Comics stylizations of 3D scenes using GPU

Jordane Suarez, Farès Belhadj and Vincent Boyer

L.I.A.S.D. Université Paris 8
2 rue de la liberté
93526 Saint-Denis Cedex, France
{suarez, amsi, boyer}@ai.univ-paris8.fr

Abstract. We propose a new comics stylization model based on a very efficient depth map generation. It is designed to render large scenes with multiple objects as well as single object in real time through a complete GPU implementation. 3D comics stylizations are generally view-dependent and only use the camera field of view to render the scene. In all cases, the depth of objects is computed according to the near and far planes while they are almost without any relation with the range depth of these objects present in the scene. Our model solves this problem by computing minimal and maximal values in the depth map. Moreover, it reproduces and improves better comics stylizations proposed for 2D images. Results show that our model is suitable for different kinds of 3D scenes and to produce various comics stylizations.

1 Introduction

Toon shading is one of the most well-known effect in Non-Photorealistic rendering. It consists in reproducing simple cartoon shading on a 3D scene and is largely used in production software or video games. Extensions and improvements have been proposed to produce different comics styles but two main problems still remain: large scenes with multiple objects are never considered and one of the fundamental effects, the color desaturation according to the object depth can be greatly improved. We propose a comics stylization model to render 3D scenes in real-time using GPU implementations. It is able to solve this two problems. A new formulation to compute depth map is proposed and is well adapted for comics stylizations of both large scene including multiple objects and single object. We also present the implementation of different comics styles. In the next section, we present previous work, our model with the depth map generation and the stylization process and finally we discuss our results.

2 Related Work

Several methods have been proposed to create comics stylizations of 3D scenes. The goal of the authors is to obtain real-time methods that can be used for example in video games. Usually, comics stylizations are only produced using a particular non-physical lighting model. The proposed lighting model and its associated shading are used to render the 3D scene.

The first method have been published by Lake et al. [1] and is available in many software solutions and called toon shading. It uses the Lambertian reflection (computing for each fragment the dot product between its normal and the normalized light direction vector pointing from the surface to the light source) as a 1D toon texture index. This technique is easily understandable, implementable, uses only diffuse reflection term but is view-independent and does not take into account the remoteness of the considered object.

Different approaches have been studied to provide depth in toon shading. Texture-based approaches use mip-mapping [2] [3] or a 2D texture [4]. Unfortunately, when the distance grows between the textured object and the viewer artifacts appear and are more visible using hatching or strokes based rendering.

Mip-maps methods proposed by Klein et al. [2] and Praun et al. [3] provide art-maps to preserve constant-sized strokes or hatching patterns at different depths and levels of lighting. Unfortunately these techniques only focus on patterns used to render the object in the scene. However, the rendering of different objects in the same scene is never studied.

The Xtoon model [4] exploits the Lambert's term on the same principle as the toon shading and uses a 2D texture index by adding a notion of details which vary according to the object depth or its orientation. As its goal is more the visual abstraction than the rendering speed, it introduces the Level Of Abstraction notion instead of classical LOD [5] [6]. Level Of Abstraction can be achieved through tone or shape details. These can be easily realized by a designer using a 2D texture. A GPU implementation is also proposed for real time renderings. But even if this process is view-dependent, it does not transmit the desired effect when the viewpoint is inside a large scene. It can be used for a single object but is not well-adapted for a large scene with multiple objects.

Other 3D methods try to improve toon shading focusing on particular effects. Anjyo et al. [7] have proposed an approach that renders comics-stylized highlights on 3D objects. Starting with an initial highlight design, using the Blinn's specular model, highlight shapes are created through several functions. Nevertheless the comics stylization for a global 3D scene is never studied.

A 2D comics stylization model has been proposed by Sauvaget et al. [8]. This method generates comics images from a single photograph using a depth map to avoid the depth-less color problem and colorizes images with a specified atmosphere. It extracts image structures from the original photograph, generates a depth map from them and finally performs a treatment on a segmented image to give a comics style. Note that the generated depth map can be enhanced by the user and different comics styles have been proposed using depth information.

This technique works well with a specific depth map but needs to be adapted to 3D scenes and even more animations.

3 Our Model

Our model is designed to render a 3D scene including multiple objects with a view-dependent comics style. Based on the stylization process described by Sauvaget et al. [8], we propose to generate a scene depth map to render it with different comics styles. We apply a stylization shader using this map and the image of the rendered scene. We convert the texture RGB information into HSV. Then, based on the depth map, we desaturate these values and, if needed, we apply an ambiance color. Moreover, we can use an edge detection algorithm as Prewitt or Laplace to finally compose the result image.

However, one of the main problems is the calculation of the appropriate depth map for a comics stylization rendering. This map is used both for the rendering and the edge detection and its computation is a crucial step. As we demonstrate hereafter, a depth map linearly or logarithmically interpolated does not give suitable results. In the following, we present our method to generate a suitable depth map and our rendering process including desaturation, ambiance color and edge stylization.

3.1 Depth Map Generation

Classical computation of the depth map consists in calculating the fragment depth according to the camera field of view (near and far plane).
It is well-known that Z-buffer is non linear:

$$depth = \frac{far + near}{far - near} + \frac{-2 \times far \times near}{Z \times (far - near)} \tag{1}$$

A better precision is obtained close to the camera near plane. Moreover, this precision is increased according to the near/far ratio (more this ratio is greater, more the Z values are densely grouped around the near plane). Thus, the Z-buffer is not convenient to computations that need an uncompressed precision along the Z axis. Unfortunately, solutions that try to interpolate the Z-buffer values fall into a depth buffer precision problem. In fact, Z-buffer values are in [0.0 ; 1.0] and have been already discretized. All following proposed solutions use the depth fragment value, hereafter noted $zfrag$.
Intuitively, a linear interpolation can be proposed as:

$$depth = 1.0 - \frac{zfrag - near}{far - near} \tag{2}$$

In that case, the depth depends on the camera field of view (near and far plane) and does not take into account the position of objects in the scene (for example, when near plane is 0.1, far plane is 1000.0 and objects in the scene are in [0.1 ; 100], depth values are around 0.9 and 1.0 for all objects) and the depth map is not suitable for any stylizations (see Figure 2).
Xtoon proposes a logarithmic interpolation :

$$depth = 1.0 - \frac{\log \frac{zfrag}{near}}{\log \frac{far}{near}} \tag{3}$$

It produces a detailed and suitable depth map only when the objects are close to the near plane. As one can see in Figure 2, the problem still remains when the objects are close to the far plane.

To address this problem, we propose to calculate the depth map according to the minimum and maximum Z values of objects in the camera field of view as follow:

1. We render the scene and store the depth of each fragment into a texture. To preserve the 32-bits depth precision, either we use a 32-bit float-valued intensity texture or each depth is stored in four components (RGBA) texels using shifting and mask operations. At the end of this step, we obtain an initial RGBA texture T_0 in which each texel contains the depth of the fragment. The texture size is (W, H) where W and H could be different but must be 2^n for some integer n;
2. Starting with $T_0(W, H)$, our shader creates a new texture $T_1(W, \frac{H}{2})$ in which we store alternatively minimal and maximal values of four texels of T_0 as follow:

   $$\forall x \in 2\mathbb{N}, \forall y \in \mathbb{N}, t^{'}(x, y) = \min_{i,j \in \{0,1\}} t(x + i, 2y + j)$$
   $$\forall x \in 2\mathbb{N} + 1, \forall y \in \mathbb{N}, t^{'}(x, y) = \max_{i,j \in \{0,1\}} t(x - i, 2y + j)$$

   Where $t^{'}$ is a texel of T_1 and t a texel of T_0 (see Figure 1).

3. As we obtain a texture $T_1(W, \frac{H}{2})$ with alternatively minimal and maximal values, we construct a texture $T_2(\frac{W}{2}, \frac{H}{4})$ in which we store alternatively the minimal and the maximal values of the four previous minimal and maximal values stored in T_1:

   $$\forall x \in 2\mathbb{N}, \forall y \in \mathbb{N}, t^{''}(x, y) = \min_{i,j \in \{0,1\}} t^{'}(2(x + i), 2y + j)$$
   $$\forall x \in 2\mathbb{N} + 1, \forall y \in \mathbb{N}, t^{''}(x, y) = \max_{i,j \in \{0,1\}} t^{'}(2(x + i) - 1, 2y + j)$$

   We repeat this process until we obtain a texture of size $(2, 1)$.

As shown in Figure 1, minimum and maximum values are finally stored in the first and the second pixels. The complexity of this algorithm is linear.

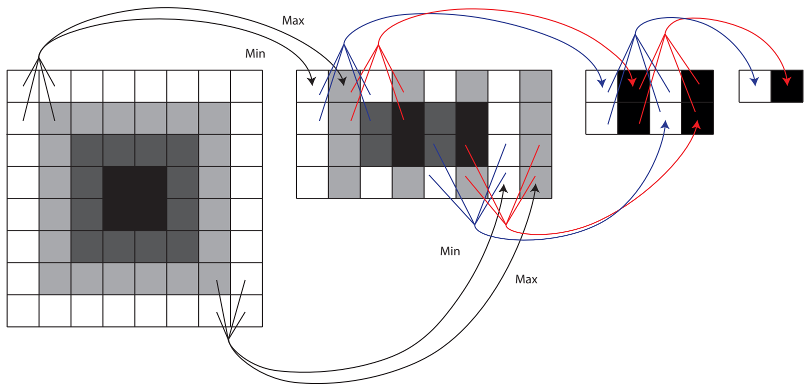

Fig. 1. Minimal and maximal depth computation from left to right: T_0, T_1, T_2 and final result.

An interpolation based on the minimum and maximum depth such as the following formula, produces a map, hereafter called minmax, covering the entire depth values.

$$depth = 1.0 - \frac{zfrag - min}{max - min} \tag{4}$$

Figure 2 presents a comparison between linear, logarithmic and minmax depth computation. On the left, we consider a scene with near plane at 0.1, far plane at 1000.0 and objects close to near plane. On the right side objects are close to the far plane. The top presents a global view of the three interpolations and the bottom shows only the object depth range. As one can see, our method ensures that produced depth map values are always uniformly distributed.

3.2 Stylization

We use the depth map previously generated to stylize our rendering. We apply desaturation, ambiance colorization and contours drawing to produce different comics stylized renderings. At this step, we have an interpolated minmax Depth Map DM and a texture T_S representing the scene. Let $t(x, y, h, s, v)$ be the texel (x, y) of T_S and $d(x, y, l)$ be the texel (x, y) of DM. We draw a quad that covers entirely the viewport to obtain one fragment f per pixel. Our rendering process is realized in the image space. Since the comics stylization do not influence the result image values:

$$f_v = t_v \tag{5}$$

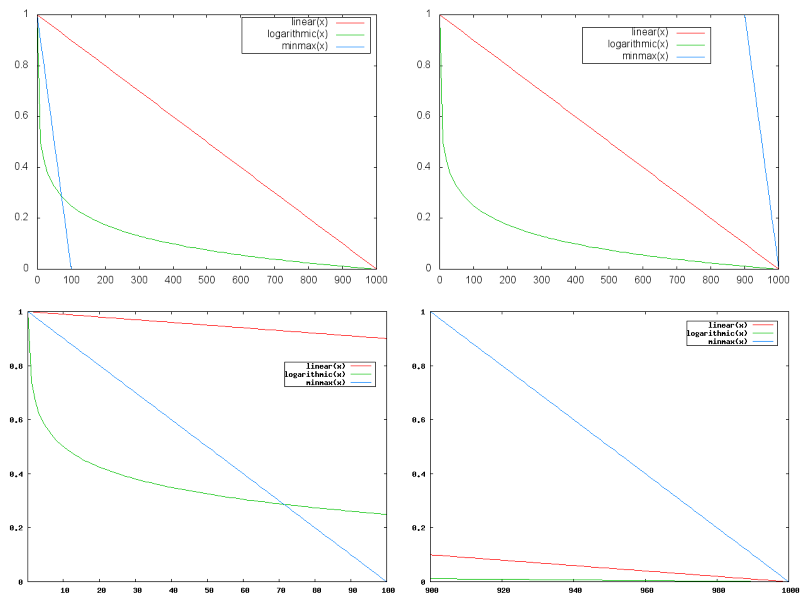

Fig. 2. Comparison of the different depth interpolation methods.

Desaturation: According to a depth map information, the desaturation is one of the most important things in comics stylization. We propose two different stylizations. The first one reproduces comics stylization proposed by Sauvaget et al. [8]:

$$f_s = t_s \times (1.0 - d_l) \tag{6}$$

To enhance the contrast with the object distance, we also propose a quadratic computation:

$$f_s = t_s \times \sqrt{1.0 - d_l} \tag{7}$$

Colorization: Four different models are proposed, two of them reproduce Sauvaget et al. method [8]. The first one follows a classical comics stylization scheme and preserve the hue in the result image:

$$f_h = t_h \tag{8}$$

The second one reproduces an atmosphere comics stylization. In that case, the user gives a hue a_h to the atmosphere and the result image is colored as:

$$f_h = a_h \tag{9}$$

As we have a high precision depth map, we are able to colorize the result image with an atmosphere according to the depth of the object. We propose a linear and a quadratic interpolation:

$$f_h = t_h \times (1.0 - d_l) + a_h \times d_l \tag{10}$$

$$f_h = t_h \times \sqrt{1.0 - d_l} + a_h \times \sqrt{d_l} \tag{11}$$

Note that the desaturation and the colorization are independent from each other and can be also combined according to user choices.

Contour drawing: We are able to enhance the result image using contours. In comics stylizations, contours are often produced to depict a small depth difference between two close objects. Therefore, according to our depth map, we implement image based solutions to create contours. Different algorithms are proposed: Prewitt and Laplace. The threshold is given by the user and contours are created when the detection algorithm finds on the DM a value greater than the threshold. This solution provides many different effects. A small threshold produces contours between close objects while a more important value creates black flat areas.

4 Results

As a preliminary result, Figure 3 presents a comparison between the linear (left), the logarithmic (center) and the minmax (right) methods. For a given scene (top of the figure), depth map produced for the three different methods (see the second line). As one can see, only our minmax depth map covers the distance object range while the other approaches take always into account near and far planes. Thus, the image produced by our method is more detailed and contrasted. We finally add a sepia atmosphere to render the scene (see the third line).

Figure 4 illustrates different stylizations realized by our model. At the first line, we use the quadratic interpolation (see equation 11), a red atmosphere to compute the resulting hue and a Laplace contour detection. We can see that, at the foreground, colors of flowers are much more preserved while the background mountains are colored with the atmosphere color. At the third line, we produce other effects including black flat areas with Prewitt algorithm and a green atmosphere. These results show that our model is more suitable than previous ones to realize comics stylization effects on large scenes. Finally, at the center-line, we demonstrate that this model is also very efficient for a single object. In that case, a quadratic desaturation (see equation 7), an atmosphere effect (see equation 9) and a Prewitt contour detection algorithm are used.

5 Conclusion

We have presented a new comics stylization model to render both single object or large scenes with multiple objects. Our model is real time, the frame rate is almost divided by 2 compared to the graphic pipeline (mountains scene is composed by one million vertices, the frame rate is 512 fps using OpenGL pipeline and 260 fps with our model using a depth map of 512 x 512 pixels). A linear depth map is computed in which the closest and farthest objects in the camera

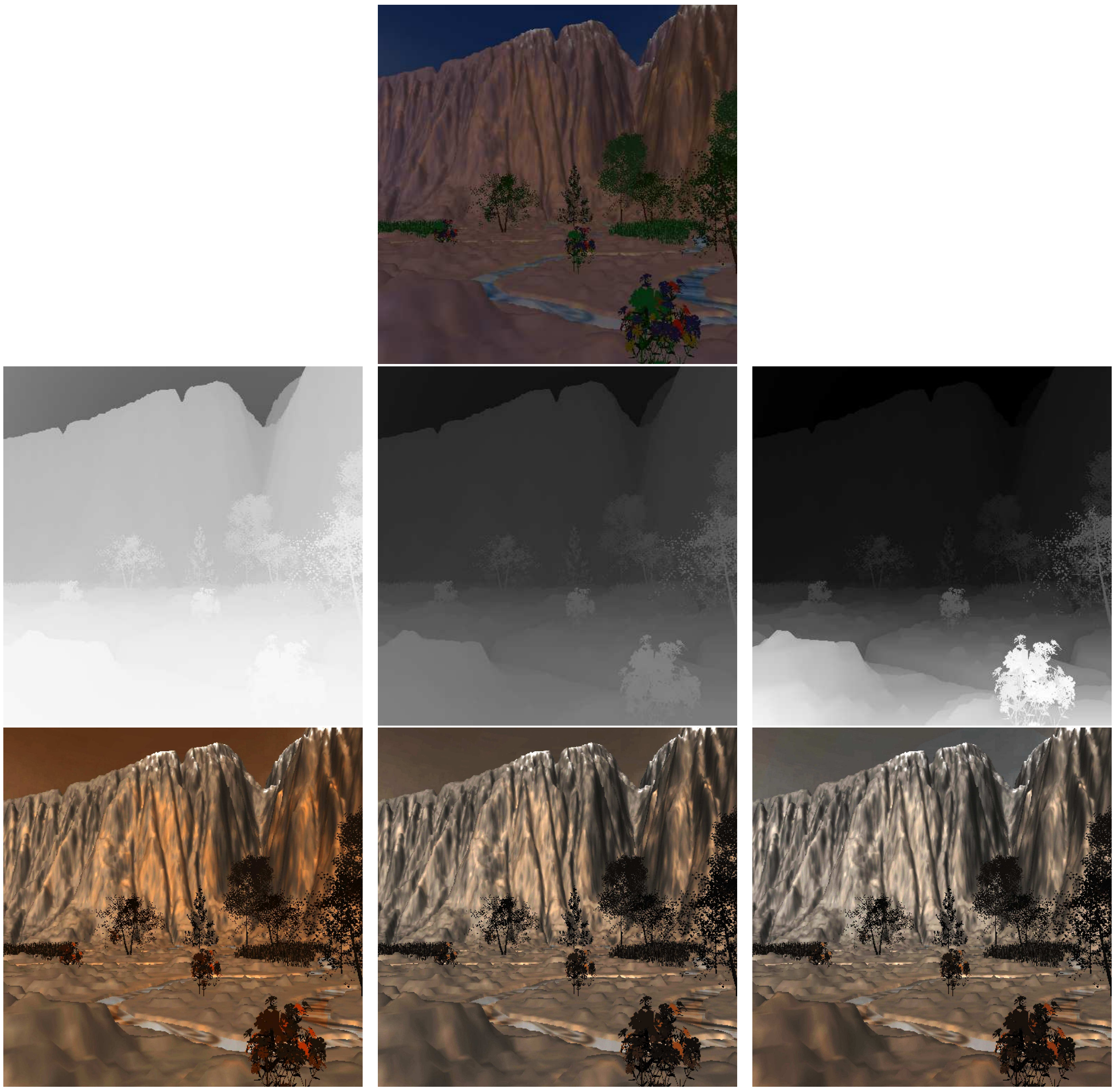

Fig. 3. Results produced using linear, logarithmic and minmax methods.

view are considered. Thus, we are able to produce a large variety of comics stylizations including desaturation, atmosphere, contours and black flat areas. Our model is suitable for any users since it is completely user-definable through a convenient GUI. As a future work, we plan to realize a treatment on the depth map to ensure temporal coherence during animation. In fact, if a closest or fare's object in the scene appears in a frame, our depth map will be affected and a

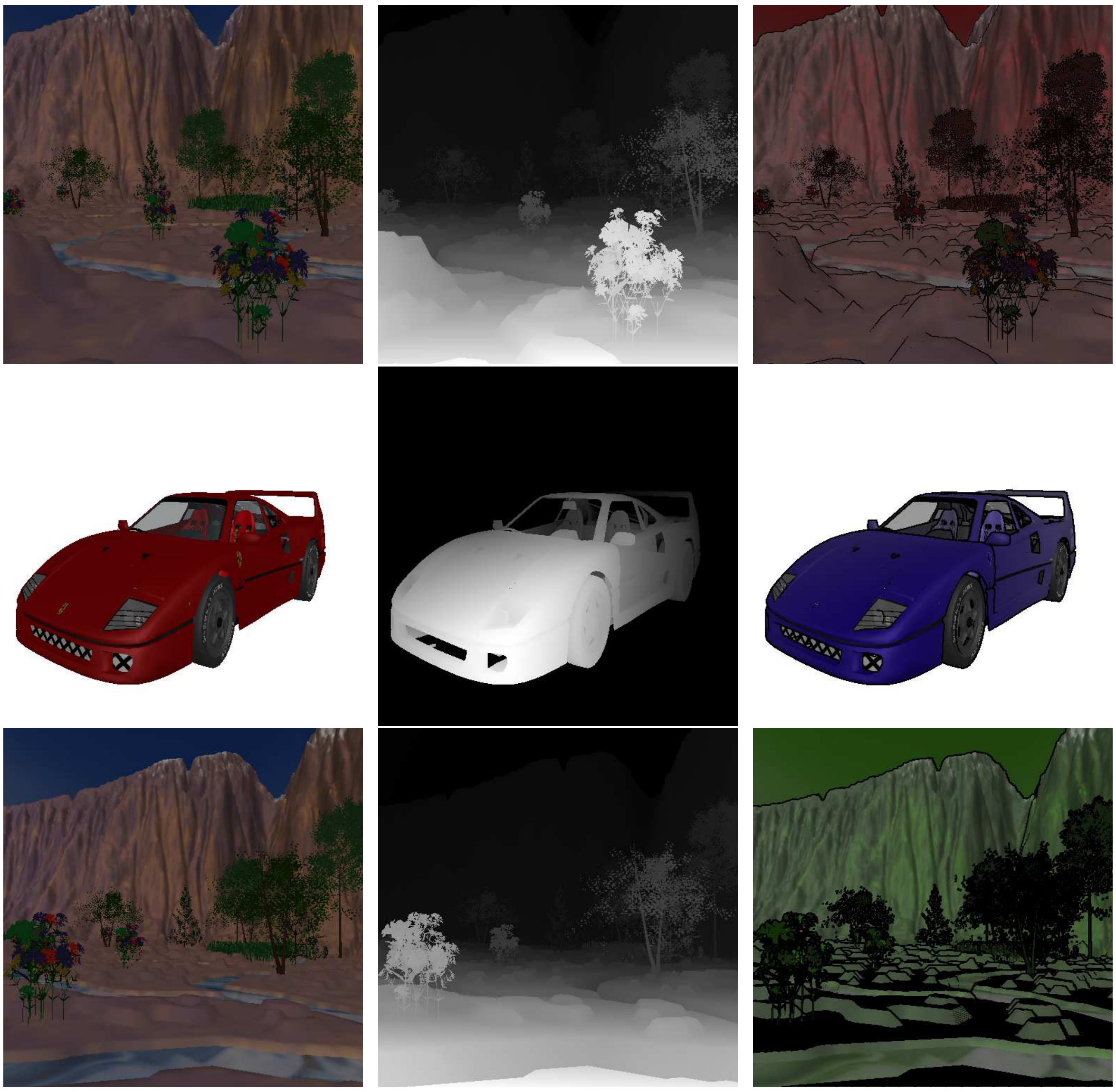

Fig. 4. Various examples realized with our model.

brutal transition will probably disturb the viewer. Moreover, we aim to improve our model integrating new styles like blurred, complementary colors or many other styles depending on the depth map.

6 Acknowledgements

This work has been performed within the Virtual Clone Studio project which is one of the *Serious Games* projects sponsored by the Ministry of Economy, Industry and Employment of the French Government.

References

1. Lake, A., Marshall, C., Harris, M., Blackstein, M.: Stylized rendering techniques for scalable real-time 3d animation. In: NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering, New York, NY, USA, ACM (2000) 13–20
2. Klein, A.W., Li, W., Kazhdan, M.M., Corrêa, W.T., Finkelstein, A., Funkhouser, T.A.: Non-photorealistic virtual environments. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (2000) 527–534
3. Ewins, J.P., Waller, M.D., White, M., Lister, P.F.: Mip-map level selection for texture mapping. IEEE Transactions on Visualization and Computer Graphics **4** (1998) 317–329
4. Barla, P., Thollot, J., Markosian, L.: X-toon: an extended toon shader. In: NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering, New York, NY, USA, ACM (2006) 127–132
5. Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N., Turner, G.A.: Real-time, continuous level of detail rendering of height fields. In: SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (1996) 109–118
6. Olano, M., Kuehne, B., Simmons, M.: Automatic shader level of detail. In: HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association (2003) 7–14
7. Anjyo, K.i., Hiramitsu, K.: Stylized highlights for cartoon rendering and animation. IEEE Comput. Graph. Appl. **23** (2003) 54–61
8. Sauvaget, C., Boyer, V.: Comics stylization from photographs. In: ISVC '08: Proceedings of the 4th International Symposium on Advances in Visual Computing, Berlin, Heidelberg, Springer-Verlag (2008) 1125–1134