Interactive lighting and shading of 2.5D models

Bruno A. D. Marques
Universidade Federal Fluminense, Brazil
Universidade Federal do ABC, Brazil
brunodortamarques@gmail.com

João Paulo Gois
Universidade Federal do ABC, Brazil
joao.gois@ufabc.edu.br

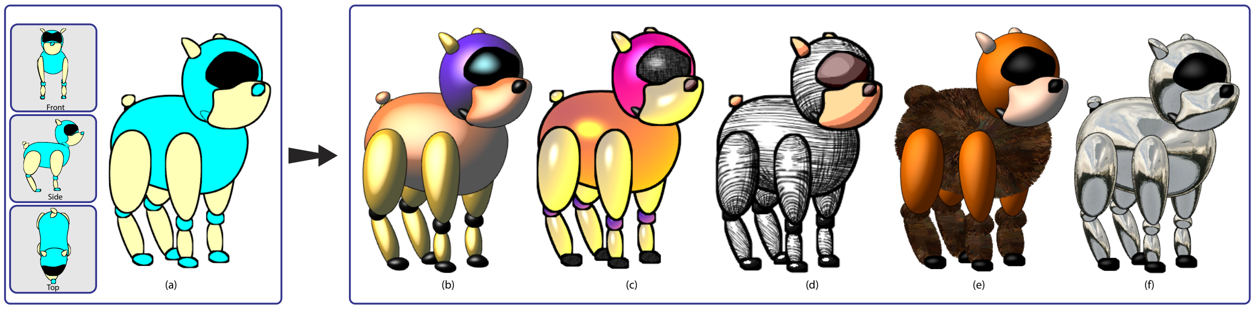

Fig. 1. Interactive lighting and shading of 2.5D models: On the left, three 2D drawings of a model and (a) the resulting 2.5D model with solid fill colors; On the right the shaded 2.5D model with (b) Phong shading (c) Gooch shading and screen-space texture hatching (d) Object-space texture hatching and cartoon shading (e) Phong shading and fur simulation (f) environment mapping and Phong shading hatching.

***Abstract*—Recent Advances in Computer assisted-methods for designing and animating 2D drawings allowed artists to achieve distinctive design styles while increasing the artist's flexibility. A advance that has gained particular attention is the 2.5D modeling, which simulates 3D transformations from a set of 2D vector arts. However, previous 2.5D modeling techniques do not allow the use of interactive lighting and shading effects. In this work we present a technique to achieve interactive 3D shading effects to 2.5D modeling[1]. Our technique relies on the graphics pipeline to infer relief and to simulate shading effects in 2.5D models in real-time. We demonstrate the application of the technique with several shading and texture effects, including: Phong, Cartoon and Gooch shadings, as well as environment mapping, fur simulation, procedural animated texture mapping and (object-space and screen-space) texture hatching. Additionally, we produced a interactive 2.5D modeling tool, wish enable users to effortless create and edit shading effects in 2.5D models.**

***Keywords*-Color; Shading; Texture; 2.5D Model;**

I. Introduction

The improvement of techniques and tools that assist artists in creating content benefit the development in all areas of visual arts. Specially, 2D animation artists have used computer-assisted tools and techniques to automate or facilitate the creation process. Strategies which generate intermediate frames between two key-frames (inbetweening techniques) [1]–[4] and simulated 3D effects in 2D drawing [5]–[8] have received much attention. In particular, 2.5D modeling techniques [4], [9] that simulate 3D transforms from a set of 2D vector art drawings. These methods generate plausible 3D points of view in any 3D angle. Three drawings are sufficient for a satisfactory simulated rotation around the 3D space [4], [9]. However, 2.5D modeling techniques include a series of limitations, in special, the inability to implement lighting and shading effects on the 2.5D model.

2D vector art drawing tools employ a wide range of shading techniques such as light mapping, gradient-based region filling, texturing and shadows. Previous to this work, interactive 2.5D modeling techniques only support 2D shapes filled with solid colors, limiting the perception of shading and lighting, as well, the ability to assign materials to the shape's surface.

Contributions: The technique presented in this work allows the user to interactively create 2.5D models with different types of materials, lighting and shading effects. We demonstrate the technique for a variety of shading effects, originally designed for real-time 3D rendering, including effects such as *Cartoon Shading*, *Phong Shading*, *Environment Mapping*, texture based Hatching and fur simulation. These effects depend on the geometric properties of 3D models such as surface normals and surface parametrization which are not present in the previous 2.5 Modeling approaches. An example of the effect achieved by our approach is shown in Figure 1 (b-f).

The technique is composed of two steps: The first step correspond to the 2.5D modeling simulation. This step is performed exclusively on the CPU. The second step generates the 2.5D shading simulation. This process is executed on the

[1]This work is based on a M.Sc. dissertation entitled ”Preenchimento e Iluminação Interativa de Modelos 2.5D”, of the Graduate Program of Computer Science of the Federal University of ABC, Brazil.

GPU where we estimate the 3D properties in the interior of 2D shapes. The use of programmable graphics hardware for the simulation of these geometrical properties allows the implementation and effects editing in interaction time. As a byproduct of our approach, we present an interactive 2.5D modeling tool capable of simulating different 3D shading effects in real-time (Sec. 5).

This work was presented and published as full paper at the *2015 Graphics Interface conference* named: *Interactive Shading of 2.5D Models* [10]. Some figures presented here were originally in this conference work.

A. Related work

In recent years, many approaches have been proposed to improve the visual appearance of 2D drawings. Di Fiore *et al.* [4] proposed an approach that aims to simulate 3D rotations from 2D drawings. The technique computes intermediate frames (inbetweening) of 2D drawings using 2.5D modeling. The artist provides a set of 2D drawings, which represents the object at different points-of-view, and defines a depth value for each curve of each 2D drawing. The technique is capable of generate new points-of-view for any 3D rotation. This is archived through the interpolation and deep ordering of the input curves. Similarly, Rivers *et al.* [9] presented the 2.5D Cartoon Model, a technique to simulate 3D rotations from 2D drawings. However, unlike previous work, the 2.5D Cartoon Model employs a automatic depth ordering of the curves. An *et al.* [11] proposed a technique to automatically convert 3D objects in 2.5D Cartoon Models. The method generates 2.5D curves and deep ordering based on the segmentation of 3D meshs.

Di Fiore and Van Reeth [12] proposed an approach to assist artists in the creation of different 2D points-of-view. The method generates 3D polygons approaching (user-input) 2D drawings. These 3D polygons serve as visual guides that help the artist to maintain the proportions and volumes consistent across multiple points-of-view.

Yeh *et al.* presented double-sided 2.5D graphics [13]. The system receives as input two images representing front and back sides of a 2D shape and then simulates geometrical effects such as roll, twist and bend. Opposing to previous 2.5D techniques, double-sided 2.5D graphics does not simulate rigid 3D rotations.

Sykora *et al.* [5] presented a technique to embed depth information into 2D drawings. The method is based on a set of depth (in)equalities that avoid the need of absolute depth values. This approach is formulated by the authors as an optimization problem that can be solved by quadratic programming algorithms. Considering that solution excessively time-consuming, they proposed an approximate solution which depends on solve a Laplace equation.

In recent years, several techniques to simulate relief from 2D drawing were presented. Some techniques rely on reconstructing the 3D geometry of objects using meshs [14]–[17], point-sets [18], or curves [19]. Other approaches tackle the issue simulating relief effects from estimated 3D normals inside a 2D curves [20]–[24]. Our approach belongs to this strategy, considering that this strategy is computationally suitable for interactive applications.

Lumo [20] is a vector-based method to shading 2D drawings. It assumes that each drawing shape belongs to the silhouette of a 3D object, implying that the normals on the curve are located in the screen plane. Lumo fills the region in the interior of the curve by propagating the normals of the curve to a 3D normal field. The propagation is achieved by an iterative dampened-spring diffuser method. Bezerra *et al.* [25] employs a similar strategy to shading 2D raster-based drawings.

Nascimento *et al.* [22] proposed an explicit method to calculate the 3D normal field in the interior of curves. Similarly, Knechtel [26] presents a structure (ancestry tree) that allows the creation, reconstruction and editing of normal maps on 2D drawings at interactive rates.

The method of Sykora et al. [5], which propagates smoothed depth information inside the drawings curves, can be used to interpolate normals and generate a 3D normal field. This technique was used in later works of the authors [7], [8]. Although these methods produces plausible results it can't be considered interactive, since it takes several seconds to generate a single frame [8].Moreover, these techniques are applied to 2D drawings and do not consider the simulation of 3D transformations.

In the next sections, we present the technical details of the interactive lighting and shading 2.5D method as well as the characteristics that we adopt in our implementation. Specifically, in Section II, we exhibit an overview of the 2.5D modeling technique. A detailed description of the technique is available at Gois *et. al* [10]. In Section III, we present the technique to lighting and shading 2.5D models. The technique is performed on CPU and GPU, each step is depict through its own subsection. In Section IV, we report our results, including an interactive 2.5D modeling tool (Section IV-A) and a brief discussion about performance of the application (Section IV-B). Conclusions, including suggestion of an application and future works for 2.5D Models are described in Section V.

II. 2.5D Modelling

The 2.5D Model uses a set of 2D drawings provided by the artist to create a model where you can simulate 3D transformations. The essential steps of the method consist of plausible interpolate shapes of 2D drawings considering their space orientation and automatic determine the depth order relation of curves that constitute the 2.5D Model.

In our approach, the user defines a set of 2D point-of-views (*e.g.* front, side, top) of the same object. The drawing in each point-of-view is composed of a set of filled shapes. A (*pitch*;*yaw*) parameter is assigned to each 2D point-of-view. These parameters represent the position of a point-of-view in the spatial orientation. The parameters have a value defined as: $-\frac{\pi}{2} \leq pitch \leq \frac{\pi}{2}$ and $-\pi \leq pitch < \pi$.

We assume that the 2D drawings for the parameters ($pitch$;yaw) = (0;0) (front view) and ($pitch$;yaw) = (0;π)

(back view) correspond to orthographic projections onto the xy-plane. We also defined a z component that can be used to determine the depth order of the drawing shape. The z component could be the z value of any non-front and non-back drawings. We take the average of the z values over all views that contain this depth information, *e.g.*, all drawings except the front and back one [10].

Rivers *et al.* [9] proposed an approach to interpolate among a set of 2D drawings. In that approach, the pitch-yaw orientation space is mapped to a 2D plane. Each 2D user input drawing is associated with a 2D position on the plane. A Delaunay triangulation is then computed considering the 2D positions of the drawings on the plane as vertices of the triangulation. New drawings are determined by barycentric interpolation inside the Delaunay triangulation. We use this method in our approach.

III. Lighting and Shading

In this section we present the technique to lighting and shading 2.5D models. The main goal of the method is to interactively simulate 3D shading effects in the interior of the shapes of the 2.5D model. These effects depend on 3D geometric properties, such as surface normals and parametrization of the surface. These properties are estimated in real time whenever a new frame is generated. Figure 2 presents an overview of the method. Bold arrows indicates processing flow and dashed arrows show data communication. The method consists of two main processing stages: the first stage is executed on CPU and the second on GPU. The CPU is responsible for the 2.5d Modeling and calculating the Contour Normals along the curve. The Contour Normals are used as for the simulation of 3D relief inside the curves. Whenever a curve is edited or created, the normals are recalculated and sent to the GPU, where the 3D relief is simulated.

A. Contour Normals

We assume that the contour of any 2D shape of a 2.5D model corresponds to the silhouette of a smooth surface (Figure 3(a)). Hence, the curve normals have components $(x, y, 0)$, where (x, y) are the normals along the contours of the 2D shape [20] [22] [10].

The Contour Normals are computed in the vertices of polylines that aproximate the contour curve(Figure 3(B)). The contour normals are computed in the CPU and stored in a Texture Buffer Object (TBO) on the GPU. Later the contour normals are propagated to the interior of the shape as a 3D normal field. This process is shown at Figure 2.

B. Dynamic Grid

Propagation of the contour normals to the interior of a shape is a task that requires intense computing power. To work around this problem, we create a dynamic grid that fits the dimensions of the bounding rectangle of the shape (Figure 3 (c)). Then we calculate normals at the vertices of the dynamic grid (Figure 3 (d)). As soon as the normals are

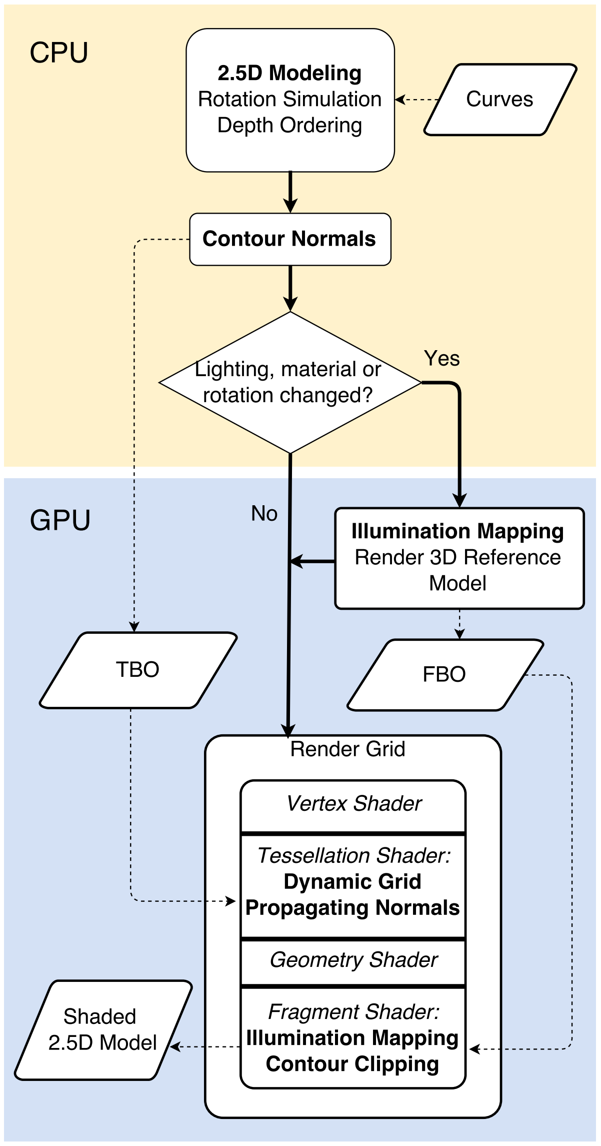

Fig. 2. Overview of the data and control flow for shading 2.5D models. The execution flows from CPU to GPU. Extracted from [10].

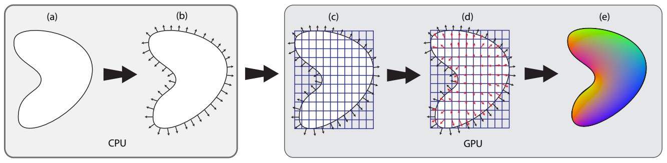

Fig. 3. The pipeline for computing normals inside a shape of a 2D curve: (a) shape interpolated by the 2.5d modeling; (b) 2D normals computed along the contour of the shape; (c) Tesselation of the bounding rectangle of the shape; (d) 3D normals estimated in the vertices of the tesselated bounding rectangle; (e) normal field interpolated for each fragment, encoded in RGB color. Extracted from [10].

calculated, they are interpolated for each fragment using the linear interpolation present in the GPU pipeline (Figure 3(e).

To create the dynamic grid in real time, we used the Tessellation Control and Tesselation evaluation shader stage included in the rendering pipeline (Figure 2). Initially the grid is composed of a single quad that is stored in a buffer object into the GPU. For each curve, this quad is appropriately scaled to fit the bounding rectangle of the curve. The levels of inner and outer tessellation are adjusted to subdivide this quad in proportion to changes in vertical and horizontal scale of the bounding rectangle. For a bounding box with different aspect ratio, the tessellation level is adjusted according to the size of most lengthy dimension.

C. Normal Field

An explicit formulation for propagating the normals was proposed by Nascimento *et al.* [22]. We used an approximation of this approach due to the fact that it can be incorporated easily into the parallel architecture of the GPU. In particular, the normal of each vertex of the dynamic grid (Section III-B) can be calculated independently in a single step in the vertex shader of the GPU. Every normal n at a vertex position p of the dynamic grid is estimated taking into account all normals $\mu_(i = 1, ..., N)$ along the curve. The x and y components of the normal n is given by:

$$n_{\{x,y\}} = \frac{\sum_{i=1}^{N} \frac{\mu_{i_{\{x,y\}}}}{\|p - p_i\|^2}}{\omega} \tag{1}$$

where

$$\omega = \sum_{i=1}^{N} \frac{1}{\|p - p_i\|^2}. \tag{2}$$

The normalization of the normal vector in p is ensured by imposing

$$n_z = \sqrt{1 - n_x^2 - n_y^2}. \tag{3}$$

We estimated a 3D normal for each vertex of the dynamic grid. The 3D normals are interpolated by the GPU raster to produce a smooth 3D Normal field. Figure 3 (e) shows the resulting interpolated normals encoded as a RGB color image.

We defined creases and ridges that can change the way light reacts to a surface. These creases affect the shading of the shape. The creases are produced from user-made markings in the interior of a 2.5D shape. These markings are cubic splines that represent shading restriction. The orientation of the normals along these markings determine whether they address a crease or a ridge on the surface of the 2.5D Model. The shading constraint are incorporated with the regular contour normals of the shape. [10]

D. Shading

Instead of calculating the lighting equations directly to each pixel within the 2D curve, we first generate an illuminated reference model having the same lighting characteristics desired in the final surface. Specifically, in this work the reference model is the rendering of a shaded sphere generated in a particular point-of-view and stored in a frame buffer object (FBO) (Figure 2). This FBO is used as a texture look-up table that is indexed by the components (x, y) of the normal vector. The choice of using an proxy model for lighting is a well known approach in the literature [20].This method accelerates the processing time of the lighting and texture mapping , since the performance of the illumination process does not depend on the complexity of the shapes.

It is important to mention that, unlike Lumo [20] which uses a static texture of the shaded sphere, in this work it is necessary to perform the rendering of this reference model whenever a change in shading or lighting occur. This is essential due to the 3D rotations in 2.5D model technique which can change the relative position between the light sources and the viewer.

E. Contour Clipping

Our approach not only consider the shading of the interior of a shape. But the entire area inside the bounding rectangle of the shape is shaded. Thus it it is important to discard the exterior of shape. For this we apply a per-pixel contour clipping operation in screen space.

Instead of performing a clipping operation on the contour of each shape, an alternative approach could be adopted. This approach would involve the creation of a mesh constrained by the interior of the shape. This approach would avoid shading of the exterior and clipping of the shape. However, this would require a high computational effort to update the VBO (vertex buffer object) wherever the shape changed. Our method requires only a single static quad as input and perform a clipping in a per-pixel accuracy.

IV. Results and Discussion

As a result of this work, we implemented a multi-platform application which includes a 2.5D modeling tool with interactive lighting and shading abilities. For this implementation we use the QT Framework version 5.3.2 [27]. All shaders have been implemented in OpenGL Shading Language (GLSL), version 4.2. All tests were performed on a computer with Intel i5-4670K configuration with 8GB RAM and GPU AMD Radeon R9 290.

A. interactive 2.5D modeling tool

A interactive 2.5D modeling tool was implemented for demonstrating our 2.5D modeling and shading technique. The application support open and closed cubic spline curves as input for the vector-art drawings. The user can navigate in 3D space from a virtual trackball or by manipulating a small view cube located in the upper right corner of the viewport.

Generally, the user draws three 2D views of the model. Being the front views, side and top the most used. These orthogonal views can be easily accessed by selecting one of the display cube sides. Other viewing angles are simply accessed freely rotating the view cube.

The editing tools are accessed by a tool bar, wish incorporate the basic features: pen, move and select tool. (Figure 4).

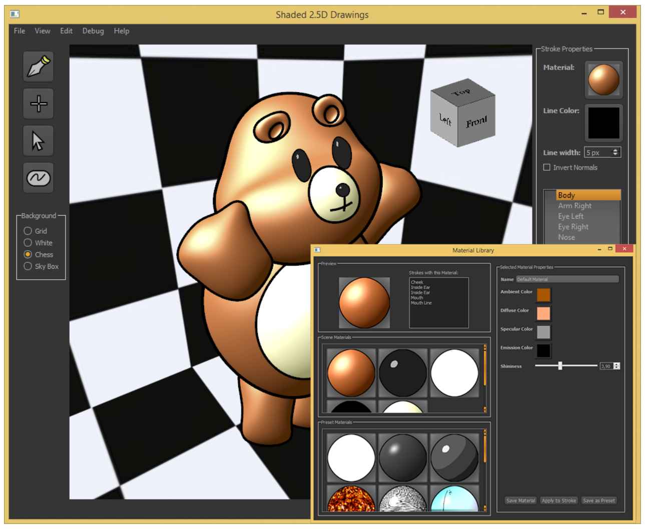

Fig. 4. The interactive 2.5D modeling tool interfaces: In the back, the main window with a Phong shaded shape; In the front, the shading and material library window. Extracted from [10]

There is also a fourth tool that allows the creation of curves that define shading constraints.

In addition to the basic tools for the creation of 2.5D model, the application also provide a interface that works as a material library containing preset shading effects. It is possible to manipulate the parameters of each effect and save them as new effects (bottom right of Figure 4).

B. Performances

TABLE I
TIMING RESULTS FOR RENDERING THE SHADED 2.5D BUNNY MODEL

(a) All shapes were rendered using the same effect.

Rendering effect	fps
Solid colors	983
Phong shading	82
Cel shading	81
Animated texture	78
Environment mapping	79
Gooch shading	81
Hatching (object-space)	80
Hatching (screen-space)	69
Fur simulation	39

(b) Number of shapes rendered with a random selected shading effect.

Shaded shapes	fps*
0	983
1	220
2	190
5	179
10	102
15	73
17	64

*The average fps for shapes rendered with random selected shading effects

We rendered the 2.5D Bunny model (Figure 5), wish is composed of 17 shapes, in a viewport scene with a resolution of 1024 × 768 pixels. Table I (a) shows the frames-per-second (FPS) where all 17 shapes were rendered using the same effect. Most of the effects reach around 80 fps. Solid colors are the most efficient. This is caused by the steps that are not present in this effect: estimation of the normals, use of 3d reference model and dynamic grid tessellation. The hatching in screen space also has a worse performance when compared to the same effect in the object space, due to the additional processing time to map the hatching textures in screen space. The fur simulation was the most time-consuming effect because it requires the geometry instancing of new primitives directly in the geometry shader.

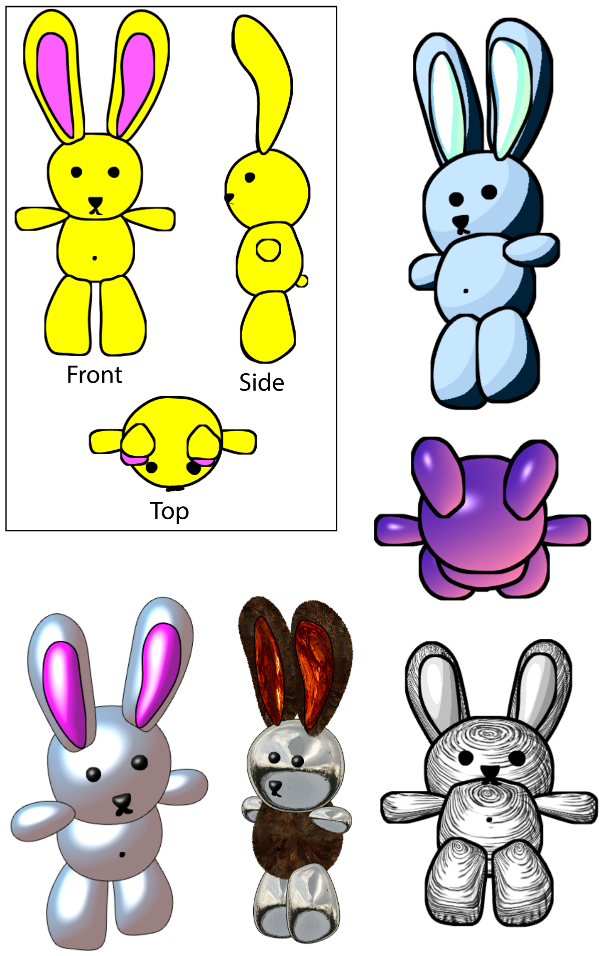

Fig. 5. The 2.5D Model Bunny. on the top left the input drawing stroke; The model shaded with a combination of cartoon shading, Gooch shading, Phong shading, fur simulation, environment mapping, animated texture shading and texture hatching.

Table I (b) presents performance results showing the impact of the number of shapes shaded with Phong shading for a 2.5D model. The first column shows the number of shapes shaded with a random selected shading effect (Phong Shading, Cartoon Shading, Gooch Shading, Texture Hatching, environment mapping or fur simulation were available) while the remaining shapes are filled with solid colors. Therefore, the use of distinct shading effects in the same model does not significantly impact the average performance.

V. CONCLUSION

We developed a technique to interactive lighting and shading 2.5D models. Prior to this work, 2.5D modeling techniques supported only shapes filled with solid colors. Our technique estimates 3D geometric properties to simulate 3D shading and lighting effects to 2.5 models.

This work was presented and published at the Graphics Interface 2015 conference as Interactive Shading of 2.5D Models [10].

As a byproduct of our technique, we implemented an interactive software tool to demonstrate different 3D shading and lighting effects in the 2.5D modeling.

Future works can tackle some limitations of the technique, wish includes: Highly concave shapes and the use of a proxy reference model for texture mapping may present distortions and aliasing artifacts. Therefor, it is necessary to investigate new texture mapping methods that fit the structure of the 2.5D model; The process of estimating the relief curves may be enhanced in order to provide methods to produce various visual effects such as simulated shadows, self-shadowing, global lighting effects, physical influence on materials (fluid simulations and gases).

An application of this work is the automatic creation of characters in 2.5D from sketches drawn by artists. In the 3D character creation process, the artist usually draws character sketches at various angles, similar to the inputs of a 2.5D model. The conversion of these drawings to 2.5D shapes would enable the construction of a 2.5D character quickly and easily.

Acknowledgment

The authors thank to Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP (proc. 2014/11067-1), the National Council of Technological and Scientific Development (CNPq), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and Universidade Federal do ABC (UFABC) for the financial support of this work.

References

[1] Z. Song, J. Yu, C. Zhou, and M. Wang, "Automatic cartoon matching in computer-assisted animation production," *Neurocomputing*, vol. 120, pp. 397–403, 2013.

[2] B. Jin and W. Geng, "Correspondence specification learned from master frames for automatic inbetweening," *Multimedia Tools and Applications*, pp. 1–17, 2014.

[3] J. Wang and J. Yu, "Correspondence construction for cartoon animation via sparse coding," in *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*. ACM, 2013, pp. 277–282.

[4] F. Di Fiore, P. Schaeken, K. Elens, and F. Van Reeth, "Automatic in-betweening in computer assisted animation by exploiting 2.5d modelling techniques," in *Computer Animation, 2001. The Fourteenth Conference on Computer Animation. Proceedings*. IEEE, 2001, pp. 192–200.

[5] D. Sykora, D. Sedlacek, S. Jinchao, J. Dingliana, and S. Collins, "Adding depth to cartoons using sparse depth (in)equalities," *Computer Graphics Forum*, vol. 29, no. 2, pp. 615–623, 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01631.x

[6] J. Lopez-Moreno, A. Bousseau, M. Agrawala, G. Drettakis *et al.*, "Depicting stylized materials with vector shade trees," *ACM Transactions on Graphics*, vol. 32, no. 4, 2013.

[7] D. Sýkora, M. Ben-Chen, M. Čadík, B. Whited, and M. Simmons, "Textoons: Practical texture mapping for hand-drawn cartoon animations," in *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, 2011, pp. 75–83.

[8] D. Sýkora, L. Kavan, M. Čadík, O. Jamriška, A. Jacobson, B. Whited, M. Simmons, and O. Sorkine-Hornung, "Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters," *ACM Transaction on Graphics*, vol. 33, no. 2, p. 16, 2014.

[9] A. Rivers, T. Igarashi, and F. Durand, "2.5d cartoon models," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 59:1–59:7, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778796

[10] J. P. Gois, B. A. D. Marques, and H. C. Batagelo, "Interactive shading of 2.5 d models," in *Proceedings of the 41st Graphics Interface Conference*. Canadian Information Processing Society, 2015, pp. 89–96. [Online]. Available: http://dl.acm.org/citation.cfm?id=2788907

[11] A. S. Fengqi An, Xiong Cai, "Automatic 2.5d cartoon modelling," in *International Conference Image and Vision Computing New Zealand*, 2011.

[12] F. Di Fiore and F. Van Reeth, "Employing approximate 3d models to enrich traditional computer assisted animation," in *Computer Animation, 2002. Proceedings of*. IEEE, 2002, pp. 183–190.

[13] C.-K. Yeh, P. Song, P.-Y. Lin, C.-W. Fu, C.-H. Lin, and T.-Y. Lee, "Double-sided 2.5d graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 2, pp. 225–235, 2013.

[14] O. A. Karpenko and J. F. Hughes, "Smoothsketch: 3d free-form shapes from complex sketches," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 589–598.

[15] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *Acm siggraph 2007 courses*. ACM, 2007, p. 21.

[16] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, no. 1, pp. 85–103, 2009.

[17] L. Olsen, F. Samavati, and J. Jorge, "Naturasketch: Modeling from images and natural sketches," *IEEE Computer Graphics and Applications*, vol. 31, no. 6, pp. 24–34, 2011.

[18] E. Brazil, I. Macedo, M. C. Sousa, L. H. de Figueiredo, and L. Velho, "Sketching variational hermite-rbf implicits," in *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*. Eurographics Association, 2010, pp. 1–8.

[19] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. McCrae, and K. Singh, "True2form: 3d curve networks from 2d sketches via selective regularization," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

[20] S. F. Johnston, "Lumo: Illumination for cel animation," in *Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering*, ser. NPAR '02. New York, NY, USA: ACM, 2002, pp. 45–ff. [Online]. Available: http://doi.acm.org/10.1145/508530.508538

[21] T.-P. Wu, C.-K. Tang, M. S. Brown, and H.-Y. Shum, "Shapepalettes: interactive normal transfer via sketching," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 44, 2007.

[22] R. Nascimento, F. Queiroz, A. Rocha, T. I. Ren, V. Mello, and A. Peixoto, "Colorization and illumination of 2d animations based on a region-tree representation," *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, vol. 0, pp. 9–16, 2011.

[23] C. Shao, A. Bousseau, A. Sheffer, and K. Singh, "Crossshade: shading concept sketches using cross-section curves," *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.

[24] J. Hahn, J. Qiu, E. Sugisaki, L. Jia, X.-C. Tai, and H. S. Seah, "Stroke-based surface reconstruction," *Numerical Mathematics: Theory, Methods and Applications*, vol. 6, no. 01, pp. 297–324, 2013.

[25] H. Bezerra, B. Feijó, and L. Velho, "An image-based shading pipeline for 2d animation," in *Proceedings...*, M. A. F. Rodrigues and A. C. Frery, Eds., Brazilian Symposium on Computer Graphics and Image Processing, 18. (SIBGRAPI). IEEE Computer Society, 2005. [Online]. Available: http://urlib.net/sid.inpe.br/banon/2005/07.12.02.02

[26] M. K. Lessa, "Construção e modificação de imagens 2d iluminadas por mapas de normais reconstruídos em tempo de interação," Master's thesis, Universidade Federal Fluminense, 2011.

[27] Qt, "Qt,project http://qt-project.org," 2015.