First publ. as: Paper / Pacific Conference on Computer Graphics and Applications (PG '04), Seoul, Korea, 2004, pp. 339-347

Non-Photorealistic Real-Time Rendering of Characteristic Faces

Thomas Luft Oliver Deussen

Department of Computer and Information Science
University of Constance, Germany
{luft, deussen}@inf.uni-konstanz.de

Abstract

We propose a system for real-time sketching of human faces. On the basis of a three-dimensional description of a face model, characteristic line strokes are extracted and represented in an artistic way. In order to enrich the results with details that cannot be determined analytically from the model, surface and anchor strokes are supplemented interactively and are maintained during animation. Because of the real-time ability of our rendering pipeline the system is suitable for interactive facial animation. Thus, interesting areas of application within the range of the virtual avatars are possible.

1 Introduction

Non-photorealistic computer graphics was established during the past few years as an independent research area. Currently there are various algorithms that are concerned with the off-line rendering of images and that simulate artistic results. Since the creation of an animation sequence that appears hand-drawn is a very complex developing process for an artist who has to draw every single frame by hand, for us an interesting area is the rendering of non-photorealistic animations and especially real-time graphics.

Aiming at a more efficient real-time rendering, this work deals particularly with the representation of face details that cannot be extracted from the geometry due to detail-limited modelling, but that are an important part of characteristic face drawings. Therefore we provide two types of lines: *surface strokes* that are applied as complete lines onto the object surface, and *anchor strokes* that are designated by anchor points on the object surface and then continued by control points positioned in 3D-space. Thus, characteristic details, such as eyes, hair, folds, and accessories can be represented in our face drawings (cf. figure 1).

For the creation of sketchy line drawings, which are strongly stylized, complex software routines are necessary.

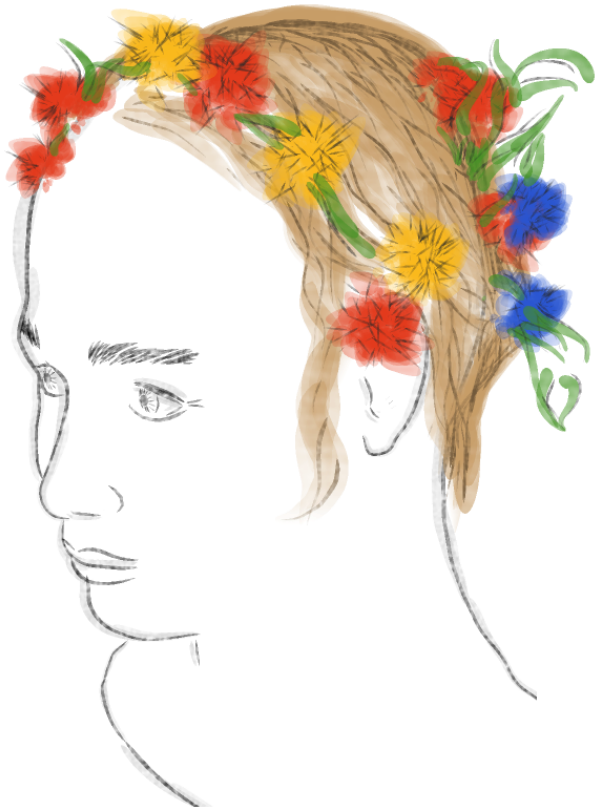


Figure 1. An example of a female head created by our system.

We present a set of algorithms and optimizations that allow the real-time rendering of scenes with moderate size. The suitability of the approach is demonstrated by a set of sample drawings.

Highly detailed line drawings of faces provide interesting perspectives for several areas of application. Especially

the real-time ability allows the creation of interactive face animation and therefore, it can be applied in chat or message systems or for the automatic generation of bearing language. Also within the range of learning software for children this form of representation is preferable, since at this age the acceptance of avatars depends strongly on their visual appearance.

2 Related Work

The creation of line drawings on basis of a 3D description of geometry was treated by many authors in the last years. Methodically image space and object space methods can be differentiated.

Image space methods can easily be computed in real-time with modern graphics hardware using pixel shaders and multi-pass rendering [16] or procedural geometry generation [19]. These procedures create simple line drawings without line stylization apart from line width and opacity variations. The reason for these limitations is the hidden surface removal on the hardware side, which principally works also for the culling of hidden lines, however partly overwrites strongly stylized lines.

For the analytic line rendering in object space various approaches were introduced that supply high-quality results, e.g. [10, 20, 24]. In [15] an object space method is presented that was optimized for the real-time rendering of scenes of moderate complexity. Also hybrid combinations of an analytic computation of the lines and a hidden line removal in image space were implemented for real-time rendering [11, 17]. In [4] the production of hand-drawn-looking animated faces is shown. However, their work is limited to two-dimensional data.

In this work we also use a hybrid approach for line rendering. In order to achieve a characteristic face representation, our approach is enhanced by the options of adding details interactively using surface strokes and anchor strokes. We introduce methods for their construction, application, and their integration into the hidden line removal algorithm.

2.1 Line Drawings of Faces

As for other objects, a very characteristic line type for a face is its silhouette line. For each frame, the computation must be executed indepently, since these lines are viewpoint dependent. In [15] a method is described that uses the local coherency of silhouettes for small viewpoint changes, and with which a better performance than using a brute force approach can be achieved. The ”srf-method” introduced by Gooch [7] is a stable and usefull method for the silhouette computation of triangulated nurbs surfaces. The procedure can also be used for faces, since faces consist of typically smooth surfaces. Another approach by DeCarlo [5] extends

Figure 2. Drawing of a male head using silhouette strokes combined with supplemented detail strokes (male A).

the silhouettes of an object by so called suggestive contours. This type of lines is located at the zero set of the radial curvature of a surface and become silhouettes when viewed from a nearby viewpoint.

In addition to silhouettes there are so called ”feature lines” needed to convey the special expressions of a face. Examples of feature lines are creases and boundaries that represent geometrical and topological discontinuity of the triangular mesh. Boundaries typically represent a part of the silhouette. Thus, it is important to draw this kind of line. Ridges and valleys are based on maximum and minimum curvatures in principal directions of the object surface. Cap and pit edges are based on concave and convex regions of the mesh and are used in the rendering system of Sousa [23]. However, finding feature lines geometrically on basis of the mesh slope or other operators often leads to unwanted results and too many or too less lines.

Our system uses the ”srf-method” [7] for the silhouette computation. For producing characteristic and lively face drawings, as is desirable for facial animation, automatically generated lines are not sufficient, as mentioned above. On the one hand it is desirable to emphasize special details of a face that are not contained in the geometry data; however,

on the other hand the real-time ability limits the modelling complexity for a face. As an example, complex hair should not be contained in the geometry data but can be sketched with a few strokes.

In the photorealistic representation, details that were not modelled are replicated by means of textures. To provide comparable results for non-photorealistic rendering, our system uses two types of lines: surface strokes and anchor strokes. These types of lines allow imitating details, like eyes, folds, hair or even glasses and other accessory, without these details having to be modelled beforehand (cf. figure 2). The surface strokes were inspired by Kalnins et al. [13] who introduced decal strokes which allow the user to draw onto the surface directly. Anchor strokes that are used for sketching hair were inspired by the graphtal strokes introduced by Kowalski et al. [14]. Contrary to our system, these strokes are not based on individual lines, but on polygonal primitives.

3 NPR Pipeline

To render and animate a complex face in real-time we have developed a non-photorealistic rendering pipeline. It consists of a preprocessing step that allows the efficient search of silhouettes. Here, the possible deformation of the model for facial animations is taken into consideration. In the next step, the computation of silhouettes as well as surface and anchor strokes is performed; a hidden line removal algorithm determines visible parts of the lines, and finally the produced lines are rendered in an artistic way.

3.1 Preprocessing

During the preprocessing step a data structure is applied that supports the efficient search of silhouettes in object space and the computation of the added detail strokes. Therefore, information about the topology of the triangular mesh such as adjacent triangles or surface normals needs to be stored. As a result of the object plasticity some restrictions arise with regard to using algorithms that would otherwise allow a more efficient silhouette searching, e.g. the gauss map [2, 7] or the normal-cone hierarchy [22].

Another task is to provide consistent surface normals when deforming the model during an animation. Thus, our system computes triangle and vertex normals on-the-fly. The vertex normals are calculated as a sum of all adjacent triangle normals. We recognized that best results for the silhouette computation are achieved if the vertex and triangle normals are not normalized.

The determination of boundary lines is also executed during the preprocessing step, because these lines are bound to the geometrical topology of the object and so they are view-independent. Only a list of concatenated control points is stored and rendered during the runtime using the actual coordinates of the mesh vertices.

3.2 Silhouette Computation

As mentioned above, our system uses here the srf-method of Gooch [7]. This procedure creates a piecewise linear silhouette approximation of a triangulated smooth surface. The silhouette edges are linearly interpolated between the mesh vertices using the dot product of the vertex normals and the view vector at the vertices. The algorithm creates either silhouette rings or partial silhouettes, beginning and ending at a boundary edge. Thereby artefacts are avoided, which occur with the computation of silhouettes based on triangle edges [11, 17].

During the silhouette determination a set of silhouette edges is produced. The search and the concatenation of these silhouette edges is performed under consideration of the triangular mesh topology. Thus, long connected partial silhouettes and silhouette rings can be found in an efficient way. In addition, the connection of silhouette edges of different objects and/or disconnected triangles is avoided, as it could happen by regarding only the image plane [17]. Finally, the control points are thinned out by summarizing those points whose Euclidean distance fall below a certain threshold value.

3.3 Hidden Line Removal

For the hidden line removal we use a depth buffer approach as presented in [11]. The depth buffer is sampled along the extracted lines and then the values are compared with those of the extracted lines. As a result, information about covered line portions are received. In order to avoid covering artefacts due to significant quantization errors of the depth map near the silhouettes, the 8-neighbourhood of a pixel is taken into consideration as proposed in [11].

In extension of this approach, we introduce two threshold values that indicate the shortest visible and invisible segments. If the length of a visible line segment falls below the first threshold it is treated as invisible and it is not drawn. Furthermore, short invisible segments below the second threshold are turned visible. As a result artefacts as the appearance of very small lines can be avoided. However, using a high threshold value, unintended covering may occur. For example, a row of actual short segments would be summarized and treated as one long covered segment.

Another problem occurs as the result of significant differences between the approximated silhouette and the original triangular mesh. As described above, the silhouette edges are interpolated within their associated triangles. There are some silhouette edges that lie on triangles facing away from the viewer. These triangles are causing covering

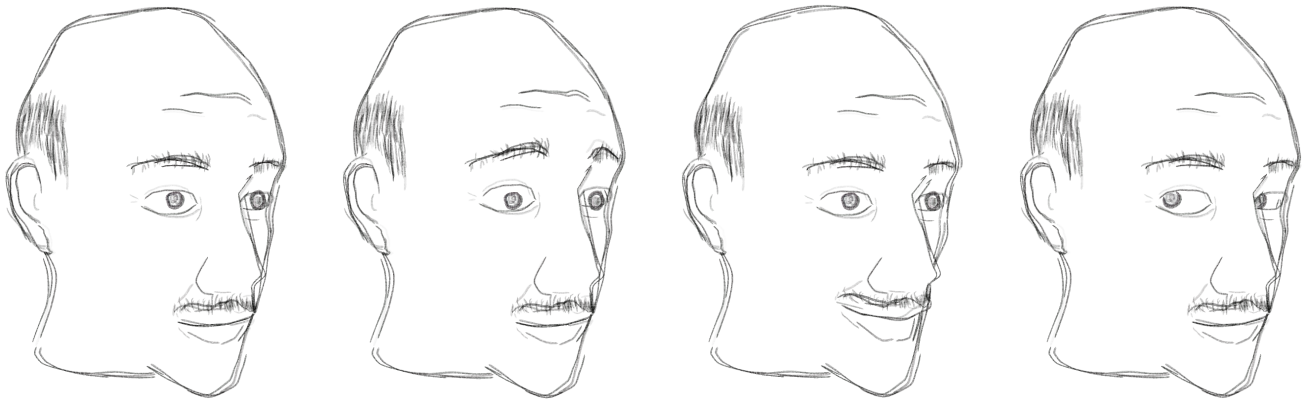


Figure 3. Properly distortion of surface strokes when deforming the underlying 3D model (male B).

artefacts, since their depth values are above those of the adjacent front faces. These artefacts are avoided by a sufficient fine triangulation or by a adaptive subdivision in regions of the silhouette.

Currently the hidden line removal is the bottle neck of the system, since reading back the depth buffer is a limiting factor for the frame rate. Nevertheless, frame rates above 30 fps can be produced on our reference system using a GeforceFX 5800 graphics board and a depth buffer resolution of 512×512 pixels (cf. table 1). A software routine for the rendering of the depth buffer may achieve a better performance on computer systems with other graphics hardware. This always depends on the topology and complexity of the mesh.

3.4 Line Generator

Finally the visible line sections are rendered by a line generator that was particularly designed for creating artistic strokes. The line path is described by the control points of the silhouette edges and approximated using cubic B-Splines. Each line possesses a certain opacity and width that can be changed continuously or stochastically along the path. Furthermore, a texture can be applied to emphasize the appearance of a particular drawing tool. The reproduction of the artistic line style is performed via the combination of different, mathematically described effects. This allows a higher degree of freedom and modelling abilitiy than approaches on the basis of example strokes given by the user [6, 9]. Each effect gives specific attributes to change the line style. For example, the effect *fragment* creates several scattert duplicates of the line path, so that the sketchy appearance of the silhouette in figure 3 can be achieved. Another effect *inaccuracy* used for almost every line style creates stochastic jittering of the line path, opacity, and width.

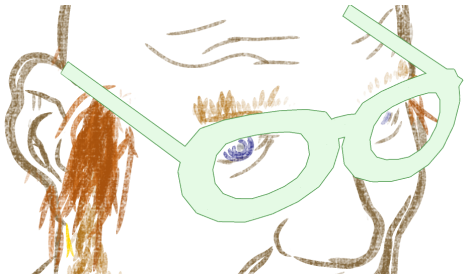
For all stochastic influences, a Perlin noise function is used [18]. Thus, the conditions for a frame-to-frame coherent representation are achieved and the effects are fully controllable. The frame-to-frame coherence plays a substantial role when viewing animated line drawings. To keep and/or to produce frame-to-frame coherent animations is not trivial especially for stylized line drawings. In [3, 12] algorithms are introduced maintaining the temporal and arc-length coherence of silhouettes. This is especially difficult due to topological changes of the silhouettes when animating the model or changing the viewpoint. Currently our system maintains temporal coherence for silhouettes only by the determinism of the used algorithms, while surface and anchor strokes are typically frame-to-frame coherent due to their static definition.

The line generator uses OpenGL quad strips for rendering stylized strokes that are generated along the underlying path. The opacity of the lines is achieved using an OpenGL blending function. In order to avoid blending artefacts at strong curvatures, each line has a constant depth value, and a blending function is used that only blends pixels with different depth values. According to the painter's algorithm the depth values of each line are increased by a small fraction.

4 Surface Strokes

Providing lines on the object surface allows the representation of various details of the object that are hard and tediuous to achieve by geometric modelling. Our system allows the user to directly draw lines on the faces similarly to [13] while preventing distortion artefacts that appear when using textures [8] in combination with perspective projection. Furthermore, in our approach the produced lines are resolution independent.

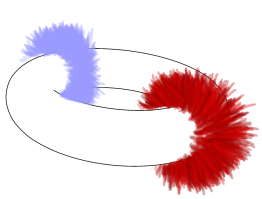
The control points of our lines are described by barycentric coordinates within the associated triangles. Only during the rendering, the Cartesian coordinates of the points are computed. The advantage of this method is that the points



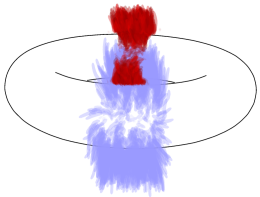
(a) generated geometry for occlusion

(b) final image with halo effect

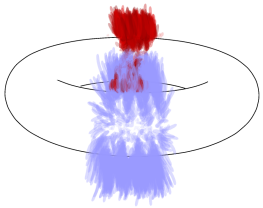
Figure 4. Using procedural geometry for occlusion and halo effect of glasses that were constructed with anchor strokes.



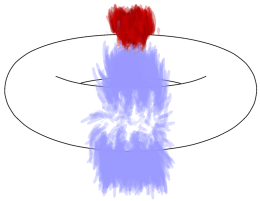
(a) an example torus with fur.



(b) worst case: All anchor strokes in the front are drawn first (28.5 fps).



(c) a compromise: Anchor strokes that have a stochastically distributed arrangement (28.5 fps).



(d) anchor strokes that are sorted by depth (18.5 fps).

Figure 5. Anchor strokes are rendered using a compromise between sorting effort and visual quality.

are properly moved during a deformation of the geometry and thus, the surface strokes are automatically deformed as well (cf. figure 3).

To compute the image space coordinates of the control points we use the already projected 2D coordinates of the mesh vertices. Thereby, the additional perspective projection of all barycentric surface points is avoided. Additionally, the actual depth value is determined for the hidden line removal, which is performed with the same algorithm that is used for the silhouettes and boundaries. With a small depth offset, the lines are shifted outward of the surface during the depth test. Thus, further covering artefacts are decreased.

There are several alternatives for the production of surface strokes. In our system an editor is integrated, which allows interactive drawing with a mouse or a pen onto the object surface. Another possibility is the extraction of edges from the original textures of the object using filters, e.g. [21]. The mapping of the texture coordinates onto the object surface can then be executed in the preprocessing step.

5 Anchor Strokes

The second additional type of strokes that is very important for our faces are anchor strokes. The underlying drawing pathes use only one surface point (the anchor) which is also described by barycentric coordinates within the associated triangle. Further control points are defined by vectors relatively to the corresponding anchor point. This way lines can be described freely in space, but are anchored on the surface of the object. An advantage of this approach is a proper movement of the anchor strokes and their orientation when the object is being deformed. In our system they are used for hair, beard, and pendants.

The hidden line removal introduced for silhouettes and boundaries can also be used. However, the hidden line removal for anchor strokes is an asynchron one. That means that anchor strokes can be occluded by other objects but they can't occlude other lines since they can't be rendered into the depth buffer. In order to achieve occlusion and line halos [1] when covering other lines, a procedural geometry creation along the anchor strokes is integrated into the depth buffer before the hidden line removal is performed (cf. figure 4). Thus, the generated geometry covers lines that are positioned behind. The geometry is a simple quad strip parallel to the projection plane. Again, a small depth offset is used to avoid covering artefacts with the associated anchor line.

In order to achieve a correct blending of the semitransparent strokes, it is important to sort the lines by their depth values according to the painters algorithm. To increase performance triangles with associated anchor points are summarized. These triangles are then sorted using an average depth value. Due to the similarity and the potentially high

(a) details are drawn onto the drawing planes

(b) drawing planes are hidden and not rendered into the depth buffer

Figure 6. Using invisible drawing planes to create accessory.

number of lines (in case of hairs) the effort of sorting can be omitted in many cases without having remarkable visual disadvantages. Therefore we use a stochastically distributed arrangement of our lines (cf. figure 5).

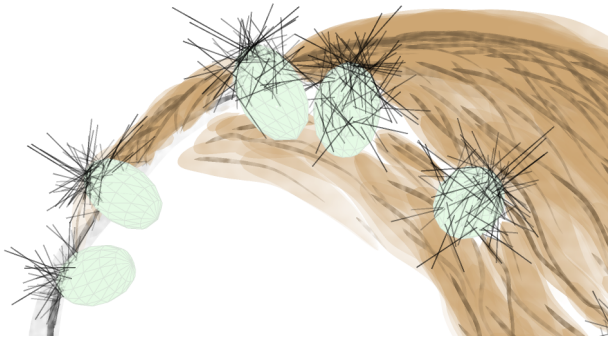
The creation of anchor strokes can be performed either by a generic function that simply creates a stochastic distribution of lines with a specific length or by a modelling tool that provides a function for the creation of these types of objects. This is especially necessary for complex line sets such as hair.

6 Modelling

As described above there are several ways to create surface and anchor strokes using interactive techniques or professional modelling tools. This section describes two techniques for the application of detail strokes and the modelling of our nonphotorealistic faces.

Comparable to bitmap textures in a photorealistic representation, ”line textures” for non-photorealistic rendering are provided. These ”textures” consist only of a number of predefined line paths. During preprocessing these descriptions of lines are mapped onto the surface using conventional texture coordinates. During rendering our line generator produces stylized surface strokes at the basis of the projected line paths. These ”line textures” are a suitable method for representing surfaces with a homogeneous structure without having to draw each individual line. This technique is applicable to both surface and anchor strokes and can be easily applied for beard and hair. Additionally, with these textures we could provide a repository of complete details such as eyebrows or eyes. Thus, the reusabilty of non-photorealistic components is given. Similar to photorealistic rendering, we are able to implement light and view dependency of these lines to provide special effects for the visualization.

Another application of surface and anchor strokes is the use of so called invisible drawing planes. This technique allows us to enhance the original scene by auxiliary objects that are carriers of detail strokes. For each carrier object the visibility and the occultation is adjustable and thus, the influence of these objects on the hidden line removal is fully controllable. If an object is visible, the silhouette of this object is drawn. If an object is covering, it is rendered into the depth buffer before performing the hidden line removal algorithm. Thus there are three meaningful combinations: an object is visible and covers other objects, an object is visible but does not cover anything or an object is invisible, but covers other objects. For example, this technique allows the user to create lines that are positioned freely in space, but are still anchored on an arbitrary object. This invisible carrier object is not rendered into the depth buffer, since occlusion of other strokes must be avoided (cf. figure 6). Another example pertaining to accessories is shown in figure 7. Here the carrier objects for the blossoms are invisible, but still cover other strokes. As a result the semitransparent blossoms are visualized without color faults that would otherwise be caused by the darker strokes of the hair.



(a) carrier objects used for the anchor strokes

(b) carrier objects turned invisible, but still covering other strokes, e.g. the hair

Figure 7. Using carrier objects and anchor strokes to create the blossoms of our female model.

7 Performance

We tested the performance of our system with a set of faces. The scenes were rendered on a Pentium 4 with 3GHz and a GeForceFX5800 graphics controller. In Table 1 the computation times of the rendering steps are compared. We used three different models with up to 35000 triangles (*#triangles*). A depth buffer size of 512×512 pixels is used for the performance test. Data preparing (*prepare*) includes the calculation time of surface normals and the perspective projection of mesh vertices. *lineComp* gives the computation time for silhouettes, surface strokes and anchor strokes. *#surface* and *#anchor* gives the number of surface and anchor strokes. The time for rendering and backreading the depth buffer is shown by *depth*. The hidden line removal (*hlr*) includes the sampling along the calculated lines and the creation of interrupt information for the line generator. Finally, the visualization of the scene using the line generator is measured (*render*). The number of OpenGL quads that are rendered in a scene is shown by *#quads*. *fps* gives the overall frames per second for the complete scene.

8 Conclusion and future work

We presented a system for the creation and rendering of characteristic drawings of faces with the appearance of hand-drawn images. The system allows the real-time rendering and animation of face models with moderate complexity. In addition to silhouette lines, two specific kinds of lines are used: surface strokes and anchor strokes. These are necessary for the non-photorealistic rendering of concrete details such as hair, eyes, and folds of our three-dimensional face models. Also it is much more complicated to model these details using the underlying mesh. Thus characteristic and highly detailed line drawings of faces can be provided in real-time. We presented different approaches for the production of these two types of lines and showed their application.

Future works aim at optimizing the system, especially when rendering anchor strokes. Currently there are still some efficiency problems such as with the rendering of complex hair. Another problem is the correct dynamic representation of hair, which requires a complex physical simulation with collision detection, and which is inadequate for our application. A highly simplified approximation using a sphere, a smaller number of hairs, and an omitting hair-to-hair interaction, can be applied in the field of real-time facial animation.

The temporal frame-to-frame coherency works well with exception of the occlusion errors and non-deterministic changes of the silhouette topology mentioned above. Here it is necessary to introduce particular algorithms that achieve a temporal frame-to-frame coherence and to avoid the temporal occlusion artefacts.

Another issue in future work is the modelling for non-photorealistic rendering, the application of the introduced carrier objects, and the detail strokes. These techniques have a profound influence on the visual appearance and the quality of the results that also include numerous applications besides the rendering of faces. For example, anchor strokes are used to create the sketched trees as shown in figure 8.

9 Acknowledgements

The female model was provided by Stefan Herz from the Filmakademie Ludwigsburg, the Male B model is courtesy of Marc Alexa (TU Darmstadt, Germany) and Wolfgang Mueller (PH Ludwigsburg, Germany).

Table 1. Computation times for the different steps of the rendering pipeline.

model	#triangles	#surface	#anchor	prepare	lineComp	depth	hlr	render	#quads	fps
male A	32085	712	1000	5.6ms	4.0ms	14.6ms	4.9ms	12.3ms	12487	24.2
female	34964	874	4032	6.1ms	5.1ms	18.3ms	7.4ms	37.1ms	21981	13.5
male B	6818	801	0	1.0ms	0.9ms	13.5ms	1.2ms	11.2ms	6498	36.1

References

[1] A. Appel, F. J. Rohlf, and A. J. Stein. The haloed line effect for hidden line elimination. In *Computer Graphics (Proceedings of SIGGRAPH 79)*, volume 13, pages 151–157, Aug. 1979.

[2] F. Benichou and G. Elber. Output sensitive extraction of silhouettes from polygonal geometry. In *Pacific Graphics '99*, Oct. 1999.

[3] L. Bourdev. Rendering nonphotorealistic strokes with temporal and arc-length coherence. Master's thesis, Brown University, 1998.

[4] I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama. Performance-driven hand-drawn animation. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 101–108. ACM Press, 2000.

[5] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848–855, July 2003.

[6] W. T. Freeman, J. B. Tenenbaum, and E. Pasztor. An example-based approach to style translation for line drawings. Technical Report TR-99-11, MERL - A Mitsubishi Electric Research Laboratory, 1999.

[7] B. Gooch, P.-P. J. Sloan, A. Gooch, P. S. Shirley, and R. Riesenfeld. Interactive technical illustration. In *1999 ACM Symposium on Interactive 3D Graphics*, pages 31–38, Apr. 1999.

[8] P. Hanrahan and P. E. Haeberli. Direct wysiwyg painting and texturing on 3d shapes. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, volume 24, pages 215–223, Aug. 1990.

[9] A. Hertzmann, N. Oliver, B. Curless, and S. M. Seitz. Curve analogies. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 233–246, June 2002.

[10] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 517–526, July 2000.

[11] T. Isenberg, N. Halper, and T. Strothotte. Stylizing Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes. *Computer Graphics Forum (Proceedings of Eurographics)*, 21(3):249–258, Sept. 2002.

[12] R. D. Kalnins, P. L. Davidson, L. Markosian, and A. Finkelstein. Coherent stylized silhouettes. *ACM Transactions on Graphics*, 22(3):856–861, July 2003.

[13] R. D. Kalnins, L. Markosian, B. J. Meier, M. A. Kowalski, J. C. Lee, P. L. Davidson, M. Webb, J. F. Hughes, and A. Finkelstein. Wysiwyg npr: drawing strokes directly on 3d models. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 755–762. ACM Press, 2002.

[14] M. A. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden, and J. Hughes. Art-based rendering of fur, grass, and trees. In A. Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 433–438, Los Angeles, 1999. Addison Wesley Longman.

[15] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, and J. F. Hughes. Real-time nonphotorealistic rendering. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 415–420, Aug. 1997.

[16] J. L. Mitchell, C. Brennan, and D. Card. Real-time image-space outlining for non-photorealistic rendering. In *Proceedings of SIGGRAPH 2002*, Sketches and Applications, page 239, 2002.

[17] J. D. Northrup and L. Markosian. Artistic silhouettes: A hybrid approach. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering*, pages 31–38, June 2000.

[18] K. Perlin. An image synthesizer. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, volume 19, pages 287–296, July 1985.

[19] R. Raskar. Hardware support for non-photorealistic rendering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 41–47. ACM Press, 2001.

[20] C. Rössl and L. Kobbelt. Line-art rendering of 3d-models. In *8th Pacific Conference on Computer Graphics and Applications*, pages 87–96, Oct. 2000.

[21] M. A. Ruzon and C. Tomasi. Color edge detection with the compass operator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 160–166, June 1999.

[22] P. V. Sander, X. Gu, S. J. Gortler, H. Hoppe, and J. Snyder. Silhouette clipping. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 327–334, July 2000.

[23] M. Sousa and P. Prusinkiewicz. A few good lines: Suggestive drawing of 3d models. *Computer Graphics Forum (Proc. of EuroGraphics '03)*, 22(3):xx–xx, 2003.

[24] G. Winkenbach and D. H. Salesin. Rendering parametric surfaces in pen and ink. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 469–476, Aug. 1996.

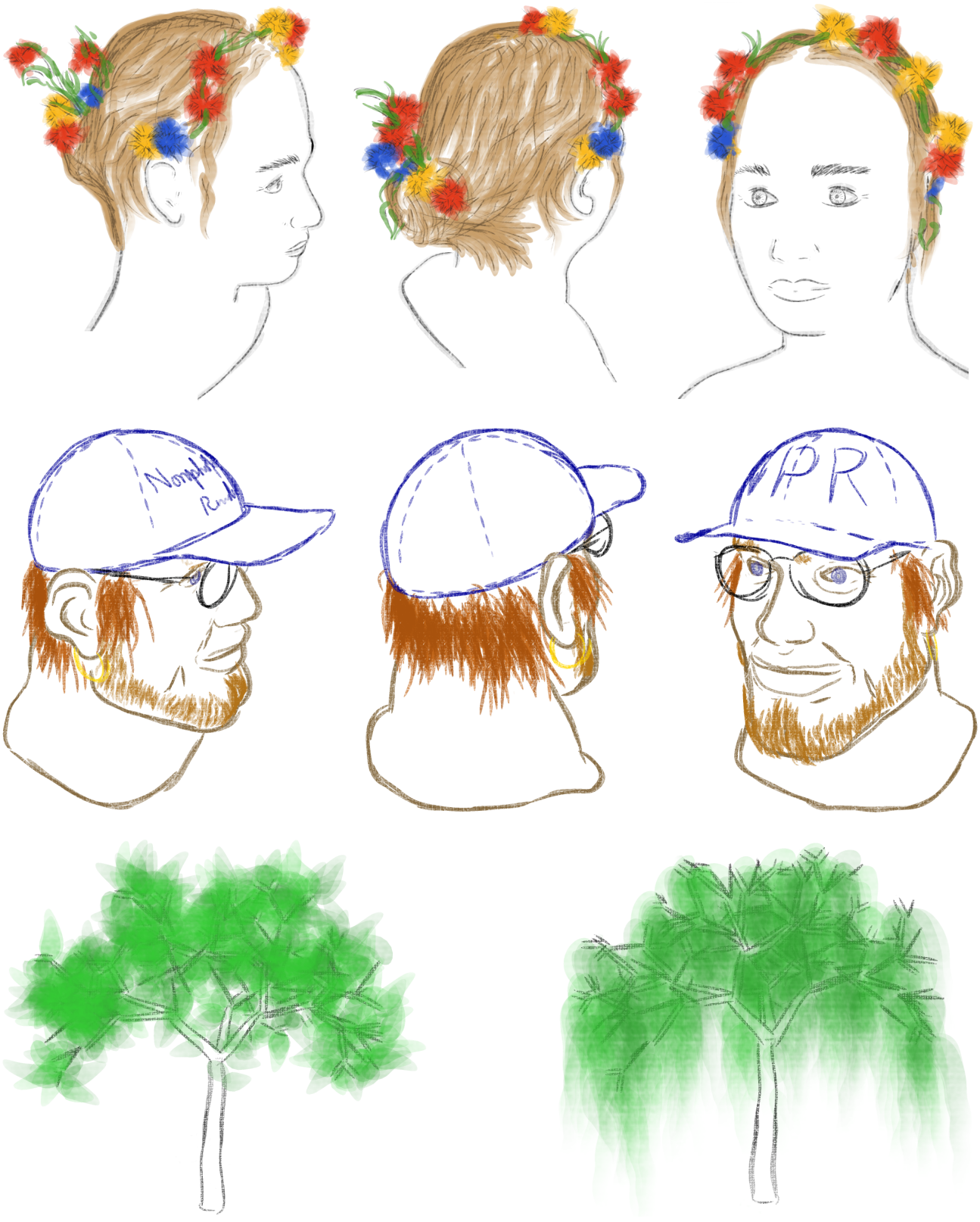


Figure 8. Different views of the male and female head and another application of our anchor strokes.