DÉGRADÉ: a New Color Shading Tool

Vincent Boyer† and Jean-Jacques Bourdin‡

† Centre de Recherche en Informatique de Lens, Université d'artois, rue de l'universite, SP 16, 62307, Lens Cedex, France, boyer@iut-lens.univ-artois.fr
‡ Laboratoire d'intelligence artificielle, Département Informatique, Université Paris 8, 2 rue de la Liberté, 93526 Saint-Denis Cedex, France, jj@ai.univ-paris8.fr

ABSTRACT

Pictures or nature rarely present uniform colored regions. Most drawings include shaded and/or color shaded surfaces. Therefore for any graphic designer the color shading tool is of major importance. The tools included in a graphics library have to respect two important conditions: they have to be fast and they have to be intuitive. Most color shading tools don't conciliate these two constraints. This paper presents a new model of color shading. Its implementation uses the fastest computation techniques up-to-date. Four examples of pictures drawn with DÉGRADÉ, the new color shading tool, are then presented.

Keywords: Shading, Texture, Two-Dimensional Graphics, Animation, Extrusion, Silhouette

1 Color shading in two dimensions

In 2D graphics the color shading tool has an important function: to fill a region of the picture with various colors and with no noticeable color separation. It is this task that is focused on in this paper. The term of color shading may also be used to denote the region filled with a color shading. In nature color shading is visible everywhere. See the delicate colors of a sky at dawn, from the red light of rising sun to hazy grey of sky through the pale pink subtly increasing. Clouds are shaded, leaves are shaded (different tones of green compose a leaf). From deep blue to cyan, the color of tropical seas is naturally color shaded.

In the art of painting the filling of a region is rarely uniform but more often color shaded. Color shading may be used to give a lighting impression (see for example the horse on Guernica [Picas37]) or a volume effect (as in the sky and stacks on Van Gogh's painting [Gogh90]) or to give a 3D impression [Vasar73] which is purely subjective since the picture is plane.

Volumizing, making an object look far-off, emphasizing its presence, present a worn out aspect, or even coloring it naturally are common tasks for any graphic designer. These tasks should be performed by the color shading tool in image synthesis.

In computer graphics color shading is mainly underestimated in its importance and usage. For example 3D graphics consider it as a native part of the global illumination process. Even in landscape pictures where the color's saturation of the farest objects should be very low, a color shading process based on the farness is almost never used. In 2D graphics the color shading tools are essential: Pixelpaint [Harri89], Photoshop [Syste98a], Powerpoint [Micro97] or Adobe Illustrator [Syste98b], present various color shading tools. But even these marketed tools should be improved.

2 Current Methods

In the paper, the following notations are used:

- A **pixel** $P(x_p, y_p, c_p)$ of the raster device is given by its coordinates x_p and y_p and its color c_p
- The **color** of a pixel $P(x_p, y_p, c_p)$ is given by the function C and $C(P) = c_p$. The color may be an entry in a Color Look-Up-Table (Color LUT) or a triplet representing the values in a color model like RGB, CMY, HLS, HSV, XYZ, YIQ, ...
- A pixel P is **connected** to a pixel Q if

$$\mid x_p - x_q \mid \in \{-1, 0, 1\},$$
$$\mid y_p - y_q \mid \in \{-1, 0, 1\} \text{ and}$$
$$\mid x_p - x_q \mid + \mid y_p - y_q \mid \neq 0$$

- A **contour** $\mathcal{C}$ ($\mathcal{C} = \{P_1, P_2, ..., P_n, P_1\}$) is a set of n ($n \in I\!N^*$) connected pixels.
- A **region** $\mathcal{R}$ is the largest connected region of pixels inside a given contour.
- A **color shading** will denote a region of the plane filled with various colors without any noticeable color separation.
- An **area** $\mathcal{A}_c = \{P \in \mathcal{R} / C(P) = c\}$ will denote an isochromatic subset of pixels. Note that these pixels may or may not be connected.

There are many different color shading methods:

- Partitioning: PixelPaint color shading tool [Harri89] partitionates the region into connected areas. The shapes of these areas are successive approximations of the region's form. For example the areas within a triangle would be triangles. Due to the discretisation the inner areas may be partially degenerated. Moreover as the partitioning process is slow it must be limited to small numbers of areas. Therefore the color separation may be clearly visible.
- Bourdin's method [Bourd90]: This method generalizes the partitioning to unconnected areas. Each area is associated to a value and each value is associated to a color (for example with a Color Look-Up-Table). The partitioning is replaced by the computation of the adequate value for each pixel of the region. The general shape of the areas is not bound to the form of the region. To reduce the expensive computing time, a class of very fast algorithms based on Discrete Differential Analysis [Brese65] was presented [Bourd90]. This method was also used to present very fast angular shading [Bourd95]. This method is therefore quick but any new shape has to be analyzed and programmed before use.
- Automatic-airbrushing: This method consists to apply a color shading as a filter on a region. The filter and the region are defined separately by the user. Once built the

filter may be reused. As mentioned by Williams [Willi91], this technique is useful in animation.

- Bi-interpolation: In Gouraud's algorithm [Goura71] a set of vertices defines a set of polygons that forms a partition of the region. Each vertex is associated to a precomputed color. Each polygon is filled by a bi-interpolation: vertical interpolation for the elements of the contour and horizontal interpolation filling the polygon line by line. The polygonal roughness of the shading implies a further use of an anti-aliasing tool.

- Subset shading: Little & Heuft's [Littl79] method is a generalization of Gouraud's. It divides a region into triangle and trapezoid subsets and fills each of them line by line. High speed is obtained by using integer arithmetic computation. This algorithm is quicker than Gouraud's but the aliasing remains visible.

3 DÉGRADÉ

The main problems of the current methods are the area or subset decomposition (for the partitioning, bi-interpolation shading and subset shading) and the programmation of any new shape (for Bourdin's method). Nobody wants to program a new shape for each new color shading and nobody conceives color shading as an area or subset decomposition. Our purpose is to produce an intuitive and fast color shading tool. This section presents a new model for color shading. This model has been used to design our tool: DÉGRADÉ was designed. It is a very fast and intuitive color shading tool.

3.1 Model

To define a region by two subsets of its contour, one has to find two sets $\mathcal{E}_1$ and $\mathcal{E}_2$ of pixels such that:

1. for each pixel P of the region there exists a pixel P_1 of $\mathcal{E}_1$ and a pixel P_2 of $\mathcal{E}_2$ such that P belongs to $[P_1, P_2]$, and

2. for each pair (P_1, P_2) of pixels of $\mathcal{E}_1 \times \mathcal{E}_2$, each pixel P from $[P_1, P_2]$ belongs to the region.

If this decomposition is not possible the region will be partitioned into smaller parts. In the following $\mathcal{E}_1$ and $\mathcal{E}_2$ are supposed given. $\mathcal{E}_1$ is a set of n_1 pixels ($\mathcal{E}_1 = \{P_1, P_2, ..., P_{n_1}\}$) and $\mathcal{E}_2$ is a set of n_2 pixels ($\mathcal{E}_2 = \{P'_1, P'_2, ..., P'_{n_2}\}$) and are called the **edges** of the region $\mathcal{R}$. It has been proved [Braqu91] that the edges have to be 4-connected.
The new algorithm consists in two steps:

- the computation of the color of each pixel P_i (and P'_j) of the edges $\mathcal{E}_1$ (and $\mathcal{E}_2$) with a curvilinear interpolation.

- the filling: for each pair of corresponding pixels (P_i, P'_j) of the edges a color shaded line is drawn from P_i to P'_j.

The next subsections will be focused on the three main problems encountered:

- The adequate non linear interpolation for the colors of each pixel of the edges.

- The correspondence of two pixels P_i and P'_j if the edges have not the same length.

- The computation of the 3D line between P_i and P'_j.

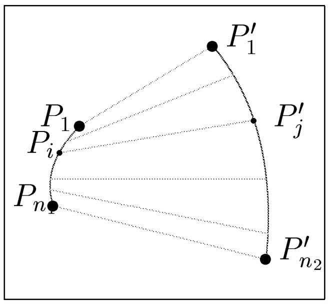

Figure 1: Our model.

3.2 Curvilinear interpolation

The determination of the color of each point of the edge can be done either manually or automatically. The manual choice may be a too long task therefore an automatic determination is more suitable.
It can be done, as in Gouraud's algorithm [Goura71] by an interpolation between the two extremities of the edge. Let $P(x_p, y_p, c_p)$ be a pixel of the edge and $Q(x_q, y_q, c_q), R(x_r, y_r, c_r)$ are the extremities of the edge.

$$c_p = \frac{d_c(P,Q) * c_r}{d_c(R,Q)} + \frac{d_c(P,R) * c_q}{d_c(R,Q)}$$

The $d_c(R,Q)$ function can be the Euclidean distance, or as in Gouraud [Goura71] the abscissae difference or a curvilinear length. As the edge is not linear a curvilinear length seems more adequate.
For example if the edge is a circle of radius ρ, let P and Q be two points of the circle, (ρ, θ_p) and (ρ, θ_q) being their polar coordinates. The circumferential distance between P and Q is:

$$d_c(P,Q) = \rho|\theta_q - \theta_p|$$

3.3 Corresponding pixels

If the edges $\mathcal{E}_1$ and $\mathcal{E}_2$ have the same number n of pixels ($n = n_1 = n_2$), for each pixel P_i of $\mathcal{E}_1$ there is one and only one corresponding pixel P'_i of $\mathcal{E}_2$. Therefore the lines to draw are (P_i, P'_i) for each $i \in [1, n]$.
If $n_1 < n_2$ then for each pixel P'_j of $\mathcal{E}_2$ there is one pixel P_i of $\mathcal{E}_1$ such that:

$$i \approx j\,\frac{n_1}{n_2}$$

For example if $n_1 = 7$ and $n_2 = 4$ the lines to draw will be (P_1, P'_1), (P_2, P'_1), (P_3, P'_2), (P_4, P'_2), (P_5, P'_3), (P_6, P'_3) and (P_7, P'_4). This computation is a linear interpolation between n_1 and n_2 and can be realized by a discrete line computation algorithm [Boyer99].

3.4 The 3D line computation

Between each pair of pixels (P_i, P'_j) the algorithm computes and draws a shaded line. If the color is given as a simple integer value (an entry in a Color Look-Up-Table) or a gray scale, a simple 3D line is computed. If the color is given in a three dimensional space (as RGB, XYZ, HSV...) the line to compute is a 5D line: from $P_i(x_i, y_i, h_i, s_i, v_i)$ to $P'_j(x'_j, y'_j, h'_j, s'_j, v'_j)$ for example. In most shadings two of these values are similar and the graphic tool keeps these values as constant. For example in figure 2 every line drawn is of constant Saturation and Hue.
Therefore DÉGRADÉ automatically adapts to compute a 2D, 3D, 4D or 5D discrete line.
The principles of fast 2D lines has been extensively studied (see [Brese65, Rokne90, Boyer99]), the fastest of these algorithms is used.
A 3D discrete line is mainly computed as projected on two axis planes [Kaufm88]. An efficient technique was used involving:

- the inner symmetry of lines to compute only half the line [Boyer99].
- properties to limit the computation to the first hexadecant.

- the double step technique as in [Rokne90, Boyer00].
- the gcd simplifications in [Angel91].
- the direct computation of the largest row of pixels (called the spans) as in [Brese85, Boyer99].
- the Discrete Differential Analysis (DDA) [Brese65].

The 4D or 5D lines are computed using a similar method.

3.5 Using DÉGRADÉ

The new technique is very fast. The computation time of each color shading is less than one second on rather old SGI Indigo. Due to the properties of the line (symmetry, spans, ...) our algorithm is faster according to the length of the line.
Examples of images produced with DÉGRADÉ are presented hereafter.
Note that in this **postscript** version, the images are degenerated. Therefore a **gif** version is attached to this document.
Note that none of these images have been anti-aliased.
• The Red Sphere (see figure 2). How can a sphere be described in 2D? There is a point of illumination P_1 with maximal light and a disk where pixels are shaded proportionally to the distance to P_1. In HSV mode the Hue and Saturation values are constant within the disk. The pixel P_1 is set to the maximal Value. The two edges are: the pixel P_1 ($\mathcal{E}_1$) and the circumference of the disk ($\mathcal{E}_2$). The Values of the pixels of $\mathcal{E}_2$ are given by their distance to P_1 and computed automatically. Then DÉGRADÉ is used and the whole disk is colored and look like a sphere.
• The Cone (see figure 3). To describe a cone one has to draw two secant lines. Their intersection is the point where the first edge is located: the top point of the cone. The other edge is the visible part of the cone base. Here also the Hue and Saturation values are constant within the figure. The Light values (in HLS) or the Value (in HSV) of the extremities of each segment to draw are identical. The Light (or Value) value increases from left to right to a maximum at one third of the edge length and then decrease constantly. The pure 2D effect makes this figure look like a cone.
• “Le Louvre” (see figure 4). The sky is drawn with an horizontal color shading. The description of the pyramid is similar to that of the cone.
• The road (see figure 5). This picture illustrates different kinds of color shading. The edges of the balloon are two bezier curves. The edges of the sphere are the point of illumination and a circle. The edges of the road are two lines. The trees are composed by two color shading. Each of them is defined by a bezier curve and a line. These examples illustrate how easy it is to use the DÉGRADÉ color shading tool. The method is very intuitive and the shape of shading easily predicted. The method is quick to be implemented in an interactive tool.

4 Conclusion

Based on a generalization of bi-linear algorithms a new method to produce color shading was presented. The DÉGRADÉ color shading tool permits to produce easily different kind of effects. This new method is also very intuitive and it is easy to produce any shape of color shading desired. It permits to realize color shading very quickly, intuitively and uses no area computation and no polygonal decomposition. Moreover the algorithms used for this method are very short, so this technique could be implemented easily in a graphic library and in hardware with a limited memory space.

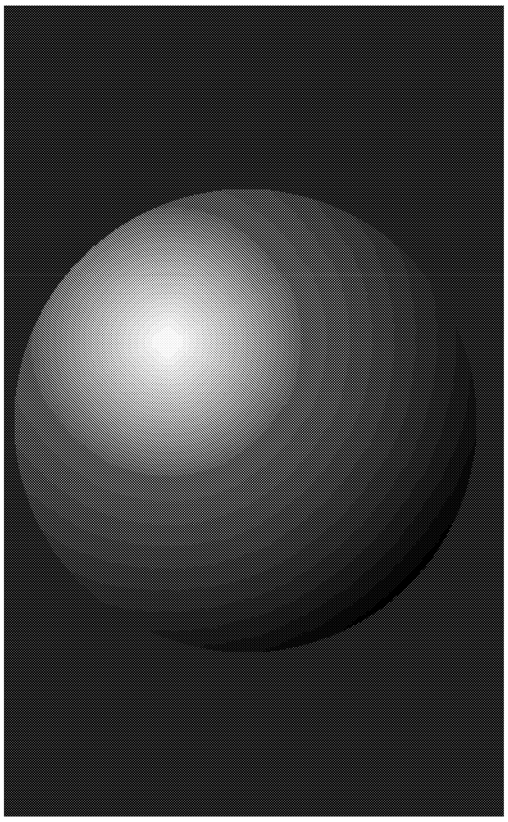
Figure 2: The Red Sphere

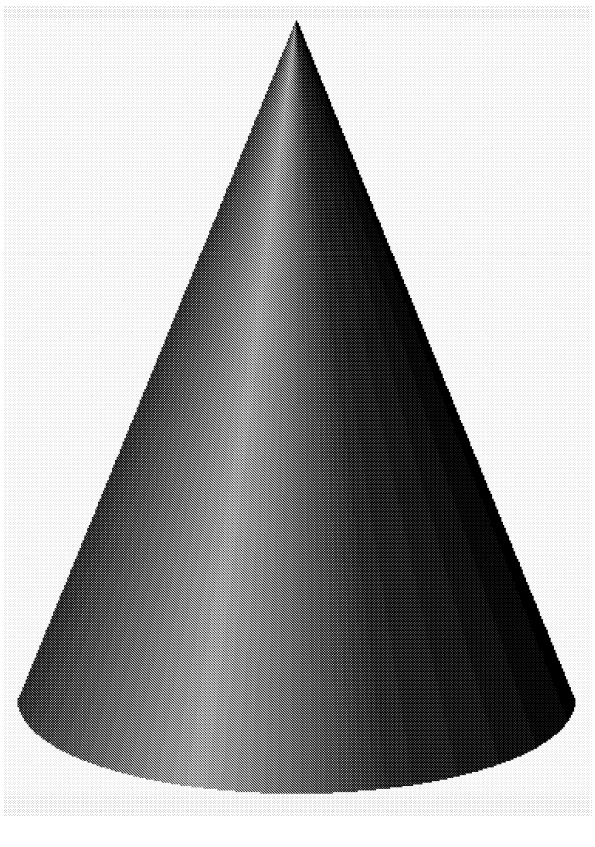
Figure 3: The Cone

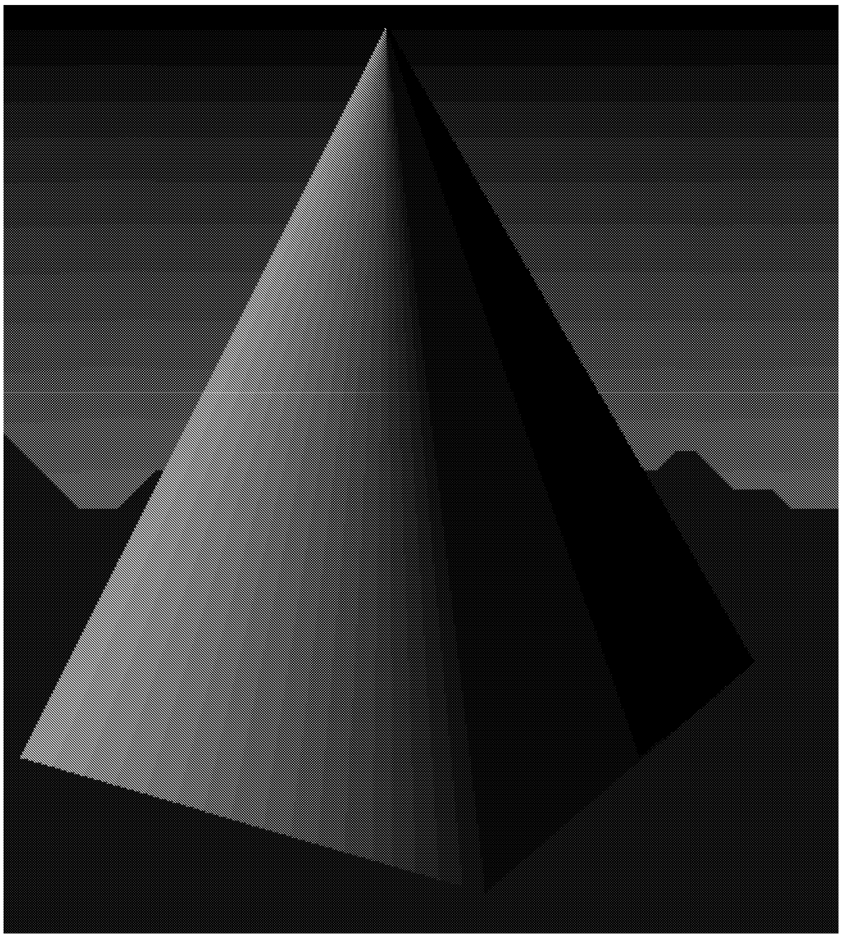

Figure 4: “Le Louvre”

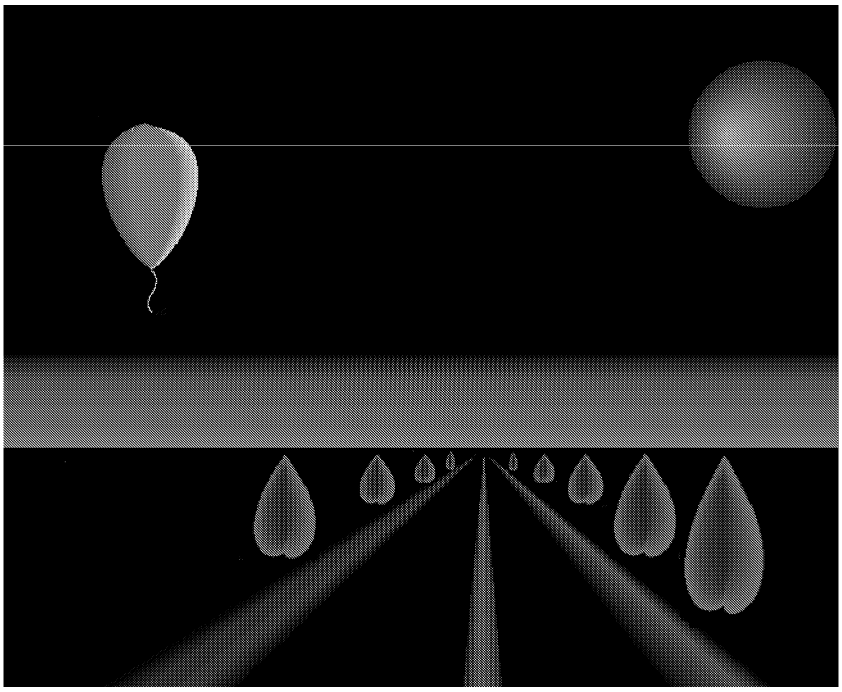

Figure 5: The road

REFERENCES

[Angel91] E. Angel and D. Morrison. Speeding Up Bresenham's Algorithm. *IEEE CG&A*, 11:16–17, November 1991.

[Bourd90] J.-J. Bourdin and J.-P. Braquelaire. Color Shading in 2D Synthesis. In *Proceedings of Eurographics'90*, pages 41–49,547–548, 1990.

[Bourd95] J.-J. Bourdin. Fast color shading. In *Winter School in Computer Graphics*, Plzen, 1995.

[Boyer99] V. Boyer and J.-J. Bourdin. Fast Lines : a Span by Span Method. In *Proceedings of Eurographics'99*, volume 18(3) of *Computer Graphics forum*. Blackwell Publishers, September 1999.

[Boyer00] V. Boyer and J.-J. Bourdin. Auto-adaptive step straight-line algorithm. *IEEE CG&A*, 20(5), 2000.

[Braqu91] J.-P. Braquelaire and P. Guitton. $2\frac{1}{2}$d scene update by insertion of contour. *Computer & Graphics*, 15(1):41–48, 1991.

[Brese65] J.E. Bresenham. Algorithm for computer control of a digital plotter. *IBM System Journal*, 4(1):25–30, 1965.

[Brese85] J.E. Bresenham. Run Length Slice Algorithm for Incremental Lines. In *Fundamental Algorithms in Computer Graphics*, pages 59–104. Springer-Verlag, 1985.

[Gogh90] V. Van Gogh. Field with Wheat Stacks, July 1890. http://www.vangoghgallery.com/painting/p_0809.htm.

[Goura71] H. Gouraud. Continuous Shading of Curved Surfaces. *IEEE Transactions on Computers*, 20(6):623–629, June 1971.

[Harri89] J. Harris, K. McGregor, J. Wolcott, and A. Samborn-Kaliczack. Pixel Paint Professional User's Manual, 1989.

[Kaufm88] A. Kaufman. Efficient Algorithms for Scan-Converting 3D Polygons. *Computer & Graphics*, 12(2):213–219, 1988.

[Littl79] W.D. Little and R. Heuft. An area shading graphics display system. *IEEE Transactions on Computers*, 28(7):528–531, July 1979.

[Micro97] Microsoft. PowerPoint 97, 1997.

[Picas37] P. Picasso. Guernica, 1937. http://museoreinasofia.mcu.es/e/colecc/Sla06/de0050.htm.

[Rokne90] J.G. Rokne, B. Wyvill, and X. Wu. Fast line scan-conversion. *ACM TOG*, 9(4):376–388, October 1990.

[Syste98a] Abobe Systems. *Adobe Photoshop 5.0 Classroom in a Book.* Abode Systems Incorporated, 1998.

[Syste98b] Adobe Systems. *Adobe Illustrator 8.0 Classroom in a Book.* Adobe Systems Incorporated, 1998.

[Vasar73] V. Vasarely. *Planetary folklore.* Chene Paris, 1973. http://www.masterworksfineart.com/inventory/vas_bio#planetary_folklore.

[Willi91] L. Williams. Shading in Two Dimensions. In *Graphics Interface'91*, pages 143–151, 1991.