Fast Colour Shading

J.J. ***BOURDIN***

Département d'Informatique
Université Paris-8
2, rue de la Liberté
93526 Saint-Denis Cédex 02—France
email : jj@aristote.univ-paris8.fr

Abstract. Colour shading a region of the plane is the action of filling its interior with continuously varying colours. Some classic methods are presented. This paper proposes an original approach to colour shading. A condition of smoothness is given. A region is modeled as a support and a chromatic function. Various shapes of shading are presented. Incremental computing and benchmarks are also discussed. In addition, we solve the problem of good visual smoothing for an angular colour shading.

Keywords: Shading, Colour shading, Two-Dimensional Graphics, Computer Graphics, Fundamental Algorithms.

1 Introduction

In the art of painting, colour shading is a very common task: the filling of a region is rarely uniform but more often shaded. In the domain of computer graphics, the term colour shading is often separated into two different domains: colour interpolation, which relates to the creation of new colours [Fishkin 83], and shading, which is a task of the visualisation, often used in 3D synthesis [Gouraud 71, Brassel 79, Little 79, Bui-Tuong 75].

As any graphist or painter, the computer graphic designer needs a natural way to perform colour shadings. In arts, shading is used to ***convey*** three dimensional form in two dimensional representation, to ***emphasise*** details and to ***add*** a feeling of distance from the viewer.

- The artist often uses the shading to create an impression of volume, of course purely subjective since the picture is a plane [Vasarely 73]. The picture produces an illusion of volume and one "sees" a sphere, a cloud, a car... In that case, within a region, pixels may have the same Hue value, but different Light values simulating the optic laws.
- To emphasise details or an object, one makes it brighter.
- To give a feeling of distance in pictures of landscapes, for example, the farthest mountains are cloudy, shaded.

For the graphic designer, colour shading is used to reproduce natural colours. The colours of the sky at dawn, the colours of a rainbow are two natural examples of colour shadings. The colour of an object is rarely uniform but mostly shaded by age, use... It is significant to

notice that colour shading is underestimated in computer graphics and that graphists usually criticise the poor colour shading functions of paintboxes.
Colour shading is then an interesting and useful problem.

2 An intuitive definition of colour shading

Intuitively, a colour shading is an area where the pixels have various colours that change without any noticeable step.

The discrete nature of the screen and images produced may create visible separations within the shaded area. In that case, the area is no longer shaded with regard to the definition. In addition, the discrete nature of the screen limits the shading: instead of a continuous change of colour, the region is partitioned into different monochromatic zones. The number of zones depends on the number of nuances of colours available on the device. On the current generation of graphic devices (Power Macintosh or Silicon Graphics Indy), only 256 levels of red, blue or green are available, and, in a colour space such like HSL, for a given Hue-Saturation association, only 256 Light levels are available. Such numbers are very few with regard to the ability of human vision: a black to white shading covering the screen generates clearly noticeable bands between the different zones, the Mach bands [Ratliff 72]. Note also that photographic slides use 700 levels of intensity [Foley 90].

3 The homogeneity condition

The subjectivity of the concepts of colours limits the writing of a general and mathematical law of smoothness: subjective distances between colours are not constants [Boynton 79, Fishkin 83, Tremeau 93], one is more able to distinguish two close yellows than two close cyans [Foley 90].

Definition: A coloured area respects the ***homogeneity condition*** if any tuple of neighbouring points has colours that are neighbours in the colour space.

We present now what this condition becomes when used in different colour spaces.

3.1. The grey scale

The first kind of colour shading is the grey scale. Colour is limited to a monochromatic function proportional to the amount of light perceived by the eyes of the viewer. The smoothness of the shading is a simple function of the ratio: $\frac{\text{number of nuances of grey}}{\text{width of the area shaded}}$.
The ideal ratio is 1, then, the average zone is of one pixel width and the shading is very smooth. If the ratio is greater than 1, some colours are not used on the device. If the ratio is smaller than 1, the zones may be visible with effect of Mach bands.

3.2. The different modes of colour shading

As stated by Fishkin [Fishkin 83], different colour spaces can be used in order to obtain colour on a screen. The most usual is the RGB space, where the colour is given by a set of three values, each value being the strength of an electron gun. With respect to our purpose, we can approximate the monitor-RGB space (mRGB space) of the screen with the mathematical RGB co-ordinates of the colour: in both spaces the problems are identical for the shading. A shading is simply an area where two points that are close on the screen have colours close in the colour space. Note also, as mentioned above, that the difference between two colours can be important while they are close in RGB space.

The distance between two colours c and c' in the RGB space can be defined using the Euclidean distance between their co-ordinates (r, g, b) and (r', g', b'). The distance is, then:

$$d(c, c') = \sqrt{(r-r')^2+(g-g')^2+(b-b')^2}$$

With that distance, the homogeneity condition becomes: a coloured area respects the homogeneity condition if every tuple ***(P, P')*** of neighbouring points has colours c_P, $c_{P'}$ with

$$d(c_P, c_{P'}) \leq 1.$$

In that case, a shading from black (0, 0, 0) to white (255, 255, 255) must use 766 steps to respect the homogeneity condition. So, an area smaller than 766 pixels can not be shaded from black to white in the RGB space with respect to this condition. The maximal value could be extended for example to 2. The condition becomes $d(c_P, c_{P'}) \leq 2$. In that case, the minimal set of a shading from black to white should be at least 256 points width (that is smooth), but a shading from green to black should be only 128 points width (that is not smooth).

Some other colour spaces are more intuitive: HSV or HSL spaces for example. In HSL space, a colour is determined by three values. The first value, the Hue value, means the dominant wavelength: it is the factor that permits us to name a colour red or green. The second value means the Saturation. Painters add various degrees of white to a given colour, in order to obtain an inverse saturation. For a red hue, a great saturation means a red colour, while a little saturation leads it to pink and then to white. The third value is the Light. Colours with zero light are black, while colours with half maximum light are red, green and so on, and colours with maximum light are white.

In this space, an area respects the homogeneity condition if any tuple of neighbouring points P, P' has colours c_P, $c_{P'}$, that respect

$$d(c_P, c_{P'}) \leq 1.$$

With this colour space one uses a shading to produce an impression of volume: the whole area will have the same chromaticity (Hue and Saturation) but different Light values, proportional to the reflection on the object.
Another purpose is also quickly achieved in that space: one is able to colour an area with a shading of colours by their chromaticity, as an object of "washed colour" or non uniformly coloured. In that case, the light component is set constant, while the hue and/or saturation changes continuously. For the painter's purpose, in both cases, the value of each component is easy to find interactively. For example, if one wants to simulate a sphere, there is a point of maximal light perceived by the viewer and there is a surface where no light comes to the eyes and that is black. The different surfaces are coloured with light decreasing from maximal at a point, to zero at the proper surface.

The limitation of this model is due to the continuity of hue values. These values follow the light spectrum and a shading from pink to purple will take intermediate colours such as yellow, green, cyan... and that may not be wanted. So HSL space is convenient for light shading, but is not "intuitive" for colour shading.

We saw that the different colour spaces may have different uses and may produce different kinds of problems or have limitations. There are many different colour spaces; each of them can be useful for a special purpose.

4 Usual techniques of colour shading

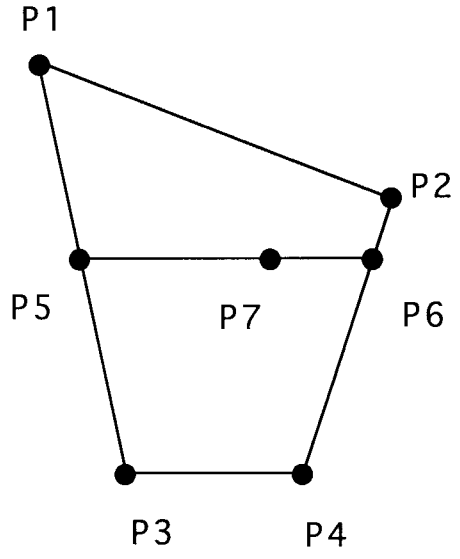
Two classical modes of shading are partitioning and Gouraud-shading. A third method will also be presented, as it generalises Gouraud shading. More recently, Williams [Williams 91] presented a method of automatic airbrushing and the realisation of shading with filters. These two methods, not easily generalisable and often slow will not be presented here.

4-1 Partitioning

Partitioning separates the area to be shaded into uniform zones. Then, each zone is coloured independently. This method permits virtually any kind of effect. But two problems persist:

- the task of partitioning may be very expensive in time if not automated,
- the forms of the zones may be partially degenerated.

In a program such as PixelPaint [Harris 89], the partitioning follows the external contour. When the form is an ellipse, the innermost forms are squares that do not look like ellipse. It should be possible to follow more precisely the outer form, at a greater computing-time expense. The obligation to produce a shading which shape is similar to the area's shape cannot be easily solved. It is a limitation as in certain cases, an elliptic shading should be wanted in a polygonal surface.



$$c_5 = c_1 * \frac{y_5 - y_3}{y_1 - y_3} + c_3 * \frac{y_5 - y_1}{y_3 - y_1}$$

$$c_7 = c_5 * \frac{x_7 - x_6}{x_5 - x_6} + c_6 * \frac{x_7 - x_5}{x_6 - x_5}$$

Figure 1: the Gouraud Shading

4-2 Gouraud Shading

In Gouraud Shading, the area is a polygon given by a set of vertices. Any vertex has a given colour. The shading is computed as bilinear: the border points receive colours proportional to the colours of the extreme vertices of the segment they belong to and to their ordinates. Then every point of the span receive colours proportional to the colours of the border points and to their abscissas.

This kind of shading produces good effects in representations of solid modelling, where the polygon can be as small as desirable and where the colour of each vertex can be computed precisely. If a car is to be designed, its colour being uniform, the colour of any point is a function of the reflection of the light on the metal. The optical laws are easy to compute, each vertex receives a colour proportional to an angular function. Two problems persist: the run time expense and the lack of smoothness. To produce a very smooth shading, one has to multiply the number of vertices and to multiply the computing time too.

That kind of shading is not of great use with paintboxes, where a graphist wants to obtain a shading on an area that may not be a polygon. It should be possible to define the shading on a polygon greater than the area and to make afterward a clipping on the area.. But the problem is that the graphist must define colours for points external to the region. External vertices might span the gamut from black to white, but the internal region will not.

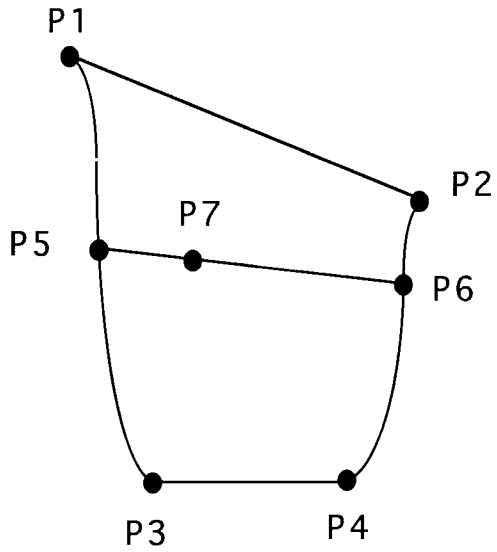
4-3 A more general method

Another method was presented by Little and Heuft [Little 79] and generalised by Royer [Royer 89]. It consists in the drawing of straight lines, not parallel to the axes. By following two paths of the contour, the contour extremes P_5 and P_6 of each line are defined. The values associated to these points are computed as proportional to the length on the contour and not to the Euclidean distance between the points. Each point P_7 of the line P_5P_6 receives a colour proportional to those of the extremities of the line and to the ratio

$\frac{d(P_7,P_6)}{d(P_5,P_6)}$. If the two path have not the same number of points, the extremities of the line will not change continuously.

Gouraud shading is a subset of this method. But important new possibilities are offered to the user. In particular, the shading can be angular: the first path on the contour is a circle and the other path is the centre of the circle.

The main problem with this method is the too long run time of the computing: the length on a contour is no trivial problem. It can be simplified by using the distance, but that is a solution of poor effects.



$$c_5 = c_3 * \frac{\overbrace{P_5\,P_1}}{\overbrace{P_3\,P_1}} + c_1 * \frac{\overbrace{P_5\,P_3}}{\overbrace{P_1\,P_3}}$$

$$c_7 = c_5 * \frac{d(P_7,P_6)}{d(P_5,P_6)} + c_6 * \frac{d(P_7,P_5)}{d(P_6,P_5)}$$

Figure 2: Generalised Shading

5 Two steps for a regular colour shading

In any case, the shading of a region will conduct to a partition of the inside pixels. Let a ***zone*** be subset of isochromatic pixels from the region to be shaded. As the plane is discrete, it is easy to associate each zone with an integer. That produces a ***support***: it is a region where each point is associated with a shading value. The support is not a coloured object but a mathematical object. The shading of a region can be done within two steps: partitioning the region into zones of different shading values, and colouring each of these zones with an adequate colour. The second step is performed by the association of each shading value with a given colour. In Colour Look-Up Table mode the integer given to each zone is the value of index associated with a triplet of values defining the current colour.

The first step produces a ***support***. It is done by an ***index function***.

The second step is the utilisation of a ***chromatic function***.

A shaded region D respects the homogeneity condition if any tuple of neighbouring points P and P' have indexes ind(P) and ind(P') that satisfy $|ind(P) - ind(P')| \leq 1$.

A region that respects the homogeneity condition is a ***homogeneous support*** (see for example figure 3). When a region is a homogeneous support, it is easy to interpolate colours in order to obtain a coloured region that looks smooth.

6 Shapes of regular colour shading

Each zone of the region may have its own shape, dependent or not on the shape of the region. For now, we will limit the problem to partition into zones that have all the same shape. If it was not the case, the regularity of the shading would be hard to maintain. The simplest shading is linear : the zones are separated by parallel lines. But other kinds of shading exist.

1	1	2	2	3
1	2	2	3	3
2	2	3	3	4
2	3	4	4	4
3	3	4	5	5

Figure 3: a homogeneous support

In order to produce a spherical volume, the artist uses an almost circular shading, where one point is at maximal light and the border is dark. The zones are concentric disks. The values of the zones are increasing with the distance between the centre and the zone. The real isochromatic colours should follow more complicated shapes, from circles to ellipses. But the concentric disks make a good spherical shading effect and that is the reason why they have been used. The general shape of such a shading is given by figure 4.

4	4	4	4	4	5	6	6
4	3	3	3	4	4	5	6
3	2	2	2	3	4	4	5
2	1	1	1	2	3	4	5
2	1	0	1	2	3	4	5
2	1	1	1	2	3	4	5
3	2	2	2	3	4	4	5

Figure 4: a circular support

7 Mathematical formulation

As the values are increasing from the centre to the border, the value of each zone is a function of the distance to the centre. While the zones are isochromatic, each pixel in the zone has the same value: the integer approximation of a multiple of the distance between the pixel and the centre.

It is possible to compute these values without any partitioning task. The region is scanned and each pixel receives a value directly from the computation of the distance.

Let $P_c\,(x_c, y_c)$ be the centre, P (x_P, y_P) a point, the index of P is given by:

$$\text{index}(P) = [\, a * \sqrt{(x_P - x_c)^2 + (y_P - y_c)^2}\,]$$

where a is the value of proportionality chosen by the user and $[x]$ is the integer approximation of the real value x.

Other shapes may be produced: ellipses, hyperbolas, ramps, etc.... The general method is to find a numeric function that gives the wanted shape. For example, here is the formula of an elliptic shading:

$$\text{index}(P) = [\, a * \sqrt{\frac{(x_P - x_c)^2}{b} + \frac{(y_P - y_c)^2}{c} + \frac{(x_P - x_c) * (y_P - y_c)}{d}}\,]$$

In both cases, the first problem is to prove that the index function respects the homogeneity condition. This depends on the constants used: if too big, two neighbouring points P and P' could have distant indices:

$$| \text{ind}(P) - \text{ind}(P') | > 1$$

If too small, the zones could be extended to maximal length, with few different zones, and then the shading obtained should make a rough impression. The constant values are to be chosen carefully in order to respect the homogeneity condition. It is also possible to use the first pass of the "region fill" to ascertain the parameters.

8 Incremental reformulating

The main problem, at this point, is the computing time. To compute the circular index of every point of the region means to compute a square root, three multiplications, three additions, one assignment and one approximation for each point of the area. This can be done automatically, but at a large computing time expense. As shown in [Bourdin 90], it is possible to use incremental computing by deducing the value of one point from the value of one of the neighbouring points. The values are calculated only in integer forms and the computation takes only three additions, two comparisons, two assignments and two shifts for every points of the area.

The principle of this incremental reformulating is to use a discrete differential analysis. An ***error function ∂(i, P)*** is used to determine the proper ***index*** value. The function ***∂*** has to be continuously increasing. Then the sign of ***∂(i, P) + ∂(i−1, P)*** determines which of ***i*** or ***i−1*** is the best approximation of the index. As the difference between the values of two neighbouring points is lesser or equal to 1, this criterion is enough to compute the value of a point from the value of one of its neighbours.

Sizes	128	256	512	1024
Sun 3	31	122	490	1959
Iris 4D	1.2	5	20	80

Run times for direct method

Sizes	128	256	512	1024
Sun 3	0.1	0.3	1.3	5.0
Iris 4D	0.1	0.3	1.3	5.2

Run times for incremental method

Table 1: run-time benchmarks

In table 1, run times are calculated for squares (length is given in pixels) and are given in seconds. Two different computers have been used for that comparison, an Iris 4D and a Sun 3. The times of direct and incremental computing are given. The support computed is a circular shading. The incremental method permits an important gain in computing times. The improvment is quite independent from the machine used.

9 A new problem: Angular Shading

The homogeneity condition has been used as a good limitation: the shadings respecting it are really smooth. But some users want a shading of particular shape that cannot respect this condition. This is the case for angular shadings. In an angular shading, a centre ***C*** and an axis ***Ax*** are defined. Each pixel ***P*** has an index proportional to the angle $\widehat{CPAx}$. The unit is chosen to make the shading unperceivable. The angular component can be quantified to ***nba*** values. The index of each point is given by the formula: $\text{index}(P)= \frac{nba}{2\prod} * \widehat{CPAx}$. For 8 values, near the centre, figure 5 presents the support produced.

The angular shading raises two main problems:

- it does not respect the homogeneity condition
- the incremental formulation of the angular functions is not trivial.

It can easily be shown that the first problem cannot be solved in the whole plane. Even with a few values, near the centre ***C***, two neighbours pixels have very different values: the difference between two diametrically opposed pixels is ***nba/2***. The homogeneity condition cannot be respected by an angular shading. But this kind of shading can be produced [Bourdin 89] without any unexpected visual effect: the shading is not theoretically smooth but no roughness can be seen.

The second problem can be solved by considering the lines separating the different zones and virtually tracing these lines with incremental computation such as Bresenham's algorithm [Bresenham 65]. The filling is done for each zone, considering that the zones have no intersection, and, moreover, that every zone are considered on each scan line. It can also be solved using the same kind of incremental reformulating mentioned above.

2	2	2	1	1	1	1	0
2	2	2	1	1	1	0	0
3	2	2	1	1	0	0	0
3	3	2	1	0	0	0	0
4	4	0	0	0	0	0	0
4	5	6	7	7	7	7	7
5	5	6	6	7	7	7	7

Figure 5: an angular support

10 Discrete Differential Analysis for angular shading

Consider a line L of slope -1 at $nba/4$ pixels from the centre I. For each pixel P we compute the intersection P_p between the line L and the line CP. The co-ordinates of the point P_p are:

$$x_p = \frac{nba * x}{4 * (x + y)} \qquad y_p = \frac{nba * y}{4 * (x + y)}$$

If P and P' are neighbouring points, let P'_p and P_p be their projections. The value of $y_{p'}$ may be computed from the knowledge of y_p. The value y_p may be used as an entry value in a table of arctangents. Hence it is called i in the followings. The error function ∂ and the addition $\Delta(i, P) = \partial(i, P) + \partial(i-1, P)$ are:

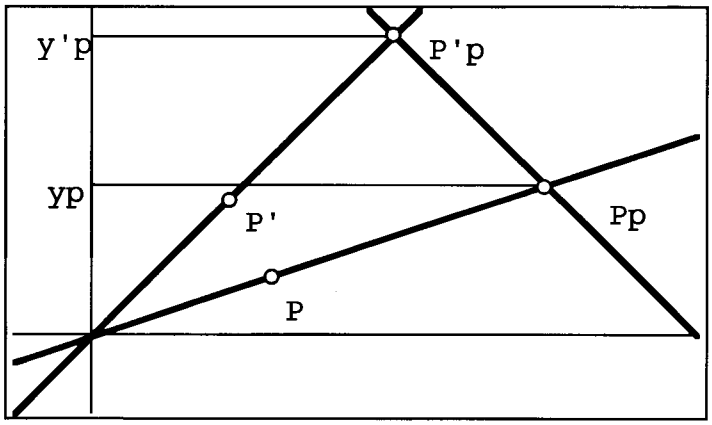


Figure 6: Angular approximation

$$\partial(i, x, y) = 4 * (x + y) * i - nba * y$$
$$\Delta(i, x, y) = 2 * (x + y) * (2 * i - 1) - nba * y$$

The incrementations of Δ are easy to deduce:

$$\Delta(i, x + dx, y + dy) = \Delta(i, x, y) + (4 * i - 2) * (dx + dy) - nba * dy$$
$$\Delta(i-1, x, y) = \Delta(i, x, y) - 4 * (x + y)$$
$$\Delta(i+1, x, y) = \Delta(i, x, y) + 4 * (x + y)$$

The complete algorithm is given in figure 7.

The benchmarks given in table 2 have been produced on a Sun Sparc and a Silicon Graphics Indigo. The sizes represent the length of a square to be shaded. The times are given in seconds. It is proved that this new algorithm is 14 times faster than the direct method.

11 Conclusion

We tried to delimit the domain of colour shading. By decomposing the shading into a support and a chromatic function it is possible to specify very easily

```
Initialisation

        input:          (xc,yc) of the centre Pc
                        (x, y) of the first point P
        output:         index i of P
                        value Δ for Δ(P,i)
{
        i = 0.5 + nba * (y - yc) / 4 * ( x - xc + y - yc )
        Δ = (x - xc + y - yc) * (4* i - 2) - nba*(y - yc)
}

Iteration step
        input:          step (dx, dy)
                        current values x, y, i et Δ
        output:         new values x, y, i et Δ
{
        Δ += ( dx + dy ) * ( 4 * i - 2 ) - 2 * nb2 * dy
        while (Δ > 0)
        {
                Δ = Δ - 4 * (x - xc + y - yc)
                i--
        }
        while ( Δ < 4 * (- x + xc - y + yc ))
        {
                Δ = Δ + 4 * (x - xc + y - yc)
                i++
        }
}
```

Figure 7 : angular shading algorithm

Sizes	*80*	*256*	*512*	*1024*	*2048*
Sun Sparc	*0.5*	*5.5*	*21.1*	*85.2*	*357*
Indigo	*0.5*	*5.1*	*20.6*	*84.4*	*350*

direct method

Sizes	*80*	*256*	*512*	*1024*	*2048*
Sun Sparc	*0.2*	*0.4*	*1.6*	*6.2*	*25*
Indigo	*0.1*	*0.4*	*1.5*	*6.1*	*24*

dda method

Table 2 : benchmark for angular shading

many different classes of shadings. The use of colour look-up tables and colour tools increases the interactivity of the task of colour shading.

The use of the homogeneity condition has been proved to insure smoothness of the shading. But the condition is not necessary, some shading that do not respect the homogeneity condition are smooth.

In the future it seems important to find another condition less restrictive but still insuring a good smoothness of the shading produced.

Acknowledgements. We would like to acknowledge the members of ***projet Lumière*** at the LaBRI (Laboratoire Bordelais de Recherche en Informatique): J.P. Braquelaire and J.P. Domenger. We thank the Siemens Nixdorf Company for their assistance.

12 Bibliography

[Bourdin 89] J.J. Bourdin, Méthodes de rendu en synthèse d'images 2D, Modèles et algorithmes, Thèse d'Université, Université Bordeaux I, 1989.

[Bourdin 90] J.J.Bourdin, J.P. Braquelaire, Color Shading in 2D Synthesis, Eurographics'90 proceedings, 41-49 and 547-548.

[Boynton 79] R.M. Boynton, ***Human Color Vision***, Holt-Rinehart-Winston 1979.

[Brassel 79] K.E. Brassel, R. Fegeas, An algorithm for shading of regions on vector display devices, Computer Graphics, Vol 13, 3, pp. 126-133, 1979.

[Bresenham 65] J.E. Bresenham, A linear algorithm for computer control of a digital plotter, IBM System Journal, Vol 4, n°1 p. 25-30, 1965.

[Bui-Tuong 75] Bui-Tuong, Phong, Illumination for computer generated pictures, CACM, 18(6), pp. 311-317, June 1975.

[Fishkin 83] K.P. Fishkin, Applying Color Science to Computer Graphics, Thesis University of California, Berkeley.

[Foley 90] J.D. Foley, A. Van Dam, S. Feiner, J. Hughes, ***Computer Graphics, Principles and Practices, second edition***, Addison Wesley.

[Gouraud 71] H. Gouraud, Continuous shading of curved surfaces, IEEE Transactions on Computers, 20(6): 623-628.

[Harris 89] J. Harris, K. McGregor, J. Wolcott, A. Samborn-Kaliczak, Pixel-Paint professional user's manual, pp. 130-136, SuperMac Technology, 1989.

[Itten 85] J. Itten, ***Art de la couleur***, Dessain & Tolra.

[Kondo 85] K. Kondo, F. Kimura, T. Tajima, An interactive rendreing technique for 3-D shapes, Eurographics'85 proceedings, 337-352 and 451-452.

[Lieberman 78] H. Lieberman, How to color in a coloring book, SIGGRAPH'78 Proceedings, CG 12(3), 1978, pp. 111-116.

[Little 79] W.D. Little, R. Heuft, An area shading graphics display system, IEEE Transactions on Computers, 28(7): 528-531.

[Ramachandran 88] V.S. Ramachandran, Perceiving shape from shading, Scientific American 259, #2, pp. 76-83, 1988.

[Ratliff 72] F. Ratliff, Contour and contrast, Scientific American, 226(6), pp.91-101, 1972.

[Royer 89] J.C. Royer, Un modèle de programmation par objets en SCHEME et application à la synthèse d'images 2D, Thèse de l'Université Bordeaux I, 1989.

[Tremeau 93] A. Tremeau, ***Contribution des modèles de la perception visuelle à l'analyse d'image couleur***, Thèse de Doctorat ès Sciences, Université Jean Monnet, Saint-Etienne, 1993.

[Vasarely 73] V. Vasarely, ***Planetary folklore***, Chêne Paris, 1973.

[Williams 91] L. Williams, Shading in two dimensions, proceedings of Graphics Interface'91, pp. 143-151.