Discrete Ray-Casting

Ivan Salam*, Philippe Nehlig*, Eric Andres

IRCOM-SIC, bât. SP2MI,
Bvd 3, Téléport 2, BP 179
86960 Futuroscope - Cedex
First_name.Last_name@sic.sp2mi.univ-poitiers.fr

Abstract. In this paper, we present a new method of discrete Ray-Casting, using parallel rays, having the property that each point of the scene belongs to one and only one ray. This study is based on the theory of arithmetic discrete geometry. The rays are modeled by Figueiredo-Reveillès' discrete naive 3D lines that tile the space and that can be incrementally generated.

Keywords : Discrete Geometry, Discrete Ray-Casting, Volume Viewing, Parallel Projection.

1 Introduction

Various fields of science and industry have become intensive users of advanced volume visualization tools. Medical imaging, for example, needs more and more precise and realistic views of anatomic volume data.

The usual 3D visualization methodology consists in the following steps: First, raw 3D data, as a result of image acquisition, is geometrically interpolated, and filtered for noise reduction, before being submitted to data segmentation algorithms in order to retain only useful information. The next step consists in selecting a visualization method amongst one of the following usual classes of visualization algorithms: Isosurface reconstruction for surface-based rendering (introduced by [15] and [18]) or voxel based rendering.

The algorithm described in this paper is a voxel based *front to back* algorithm, i.e. we scan the data space from the nearest voxel to the farthest. This class of volume rendering algorithms has been introduced by [9]. The theoretical background to our algorithm is fundamental results from *Arithmetic Discrete Geometry*, mainly due to [17]. The originality of this approach is the exclusive and intensive use of integer arithmetic which allows to handle a large amount of data faster than usual methods. The discrete approach has already been used in the past [19,14], its drawbacks and advantages have been analyzed in [5,6].

The originality of our algorithm is the use of Figueiredo-Reveillès [7] discrete lines, which are the thinnest possible 26-connected lines, that allow, through a 3D tessellating theorem (new result), to reach all the *visible* voxels exactly once, without losses or doublings.
The resulting discrete ray casting algorithm, can be sketched informally as:

* Most of this development has been achieved at the LSIIT, ULP, Strasbourg.

G. Bertrand, M. Couprie, L. Perroton (Eds.): DGCI'99, LNCS 1568, pp. 435–446, 1999.
© Springer-Verlag Berlin Heidelberg 1999

```
For each intermediate screen pixel, follow the corresponding
(if available) 3D ray inside of the data space from the front
towards the back. If the back side is reached, then give the
pixel a background color, else, if the object is touched, give
the pixel the corresponding object color.
```

Our algorithm, as well as all other discrete ray casting algorithms, is characterized by the fact that the complexity in time *does not* depend on the complexity of the scene, but only on the number of *empty* voxels to be scanned before reaching visible object voxels. This because there is no need to use heavily time-expensive intersection algorithms for each ray and for each object. The ray is stopped as soon as it reaches the object, the collision test consists in testing the voxel value.

The paper is organized as follows. In section 2 and 3 we recall fundamental definitions of discrete geometry. Section 4 concentrates on the actual discrete ray-casting algorithm.

2 Basic Definitions

In this section, we recall some basic definitions from discrete geometry. They will be useful further on. Arithmetic Discrete Geometry has been introduced and studied in [17,1,13]. Discrete planes have been studied in [8,1,2].

The *Discrete coordinate plane* consists of unit squares, called *pixels*, centered on the integer points of the two-dimensional Cartesian coordinate system in the plane.

The *Discrete coordinate space* consists of unit cubes, called *voxels*, centered on the integer points of the three-dimensional Cartesian coordinate system in the space. The pixels / voxels coordinates are the coordinates of their centers.

Definition 1 (2D adjacency). *Two pixels are* 4-connected *if they have a common side. Two pixels are 8-connected if they share a common vertex or side.*

Definition 2 (3D adjacency). *Two voxels are* 6-connected *if they have a common face. Two voxels are 18-connected if they share a common edge or face. Two voxels are 26-connected if they share a common vertex, edge or face.*

Definition 3 (neighborhood). *The set of all pixels adjacent to a given pixel P is called the* neighborhood of P.

Following definitions (4 and 1) of a 2D discrete line are equivalent.

Definition 4 ([17]). *A* 2D discrete line *is a set of pixels* $L(a,b,c,\omega) = \{(x,y) \in \mathrm{Z}^2 \,|0 \leq ax+by+c < \omega\}$, *where* $a,b,c \in \mathrm{Z}$ *and* $\omega \in \mathrm{N}^\star$. ω *is called the* arithmetical thickness *of the discrete line and* c *the* translation constant.

Result 1 ([17]) *A pixel* $(x,y) \in \mathrm{Z}^2$ *belongs to the 2D discrete line* $L(a,b,c,\omega)$ *if* $\left[\frac{ax+by+c}{\omega}\right] = 0$, *where* $[r]$, *with* $r \in \mathrm{R}$, *denotes the greatest integer not exceeding* r.

Definition 5 (run). *Along a 2D discrete line, functionnal over the X-coordinate, a set of consecutive pixels of same Y-coordinate is called a* run.

Using Result 1, it is easy to define the set L_k, the discrete line parallel to $L(a,b,c,\omega)$, by $\left\{(x,y) \in \mathrm{Z}^2 \,\middle|\, \left\lfloor \frac{ax+by+c}{\omega} \right\rfloor = k \right\}$, where $k \in \mathrm{Z}$.

Proposition 1 ([17]). *The set of all the L_k tiles the plane.*

Definition 6 (naive line, [17]). *When $\omega = \max(|a|,|b|)$, the 2D discrete line $L(a,b,c,\omega)$ is called a* naive *line.*

Proposition 2 ([17]). *A naive line is 8-connected minimal (i.e. if one removes a single pixel, the line is no more 8-connected).*

A special case of a naive line which appears to be one of the best discrete approximation to the corresponding continuous straight line among the set of all possible naive lines is the *Bresenham line* [4]. The Bresenham line was first described algorithmically, but we now have an analytical definition [17]. Let $L(a,b,c,\omega)$ be a naive line, passing through $O(0,0)$, with $c = \left\lfloor \frac{\omega}{2} \right\rfloor$, then this is a Bresenham line.

As a Bresenham line $L(a,b,c,\omega)$ tiles the plane, L_k goes through any point (x,y) of the plane, where $k = \left\lfloor \frac{ax+by+c}{\omega} \right\rfloor = K_L(x,y)$. A discrete line is *functional* over the Ox (resp. Oy) coordinate axis if there corresponds one and only one pixel to any given coordinate x (resp. y) on this line.

Proposition 3 ([3]). *A 2D naive line is functional over the coordinate axis that makes an angle less than or equal to* 45 *degrees.*

Proposition 4. *Let $L_k(a,b,c,\omega)$ be a naive line functional over X and $b > 0$. Given a coordinate x over X, there is only one pixel (x,y) belonging to L_k: $y = Y(L_k,x)$, where $Y(L_k,x) = -\left\lfloor \frac{ax+c-kb}{b} \right\rfloor$.*

Similarly to the definition of discrete lines, following definitions (7 and 2) of a discrete plane are equivalent.

Definition 7 (discrete plane, [17]). *A* discrete plane *is a set of voxels $P(a,b,c,d,\omega) = \left\{(x,y,z) \in \mathrm{Z}^3 \,|\, 0 \leq ax+by+cz+d < \omega \right\}$ where $a,b,c,d \in \mathrm{Z}$ and $\omega \in \mathrm{N}^\star$. ω is called the* arithmetical thickness *of the discrete plane and d is a* translation constant. *The vector (a,b,c) is a* normal *vector to the plane P.*

Result 2 (discrete plane, [17]) *A voxel $(x,y,z) \in \mathrm{Z}^3$ belongs to the discrete plane $P(a,b,c,d,\omega)$ if $\left\lfloor \frac{ax+by+cz+d}{\omega} \right\rfloor = 0$.*

Using Result 2 it is easy to define the set P_k, the discrete plane parallel to $P(a,b,c,d,\omega)$, by $\left\{(x,y,z) \in \mathrm{Z}^3 \,\middle|\, \left\lfloor \frac{ax+by+cz+d}{\omega} \right\rfloor = k \right\}$, where $k \in \mathrm{Z}$.

Following definitions and results are due to [1].

Proposition 5. *The set of all the P_k tiles the space.*

Definition 8 (tunnels through a discrete plane). *The plane $P(a, b, c, d, \omega)$ has a k-tunnel (k = 6, 18, 26) if there exist two k-adjacent voxels $A(x_A, y_A, z_A)$ and $B(x_B, y_B, z_B)$ such that $ax_A+by_A+cz_A+d < 0$ and $ax_B+by_B+cz_B+d \geq \omega$.*

Definition 9 (naive plane, [1,2]). *When $\omega = \max(|a|, |b|, |c|)$, the discrete plane is called a* naive *plane.*

Proposition 6 ([1,2]). *A naive plane is 18-connected. This is the smallest value of ω for which there is no 6-tunnels.*

Proposition 7. *A naive plane is functional over at least one of the coordinate planes Oxy, Oxz, or Oyz.*

3 Discrete Naive 3D Lines

In this section we show some important new properties based on the definition of a discrete naive 3D line given by Figueiredo-Reveillès [7].

Their discrete naive 3D line is the subset of Z^3 defined by the intersection of two planes obtained by extrusion of two discrete Bresenham 2D lines. The properties of such a discrete naive 3D line are the following ones:

- 26-connected.
- minimal (i.e. if one removes a single voxel, the line is no more 26-connected).
- functional over (at least) 2 planes within Oxy, Oxz, or Oyz.

Using this definition, the problem of generating a discrete 3D line is reduced to two 2D problems, since the selected discrete 3D line is characterized by two discrete 2D Bresenham lines: D_1 and D_2

Theorem 1. *The two 2D Bresenham lines characterizing the naive 3D line are functional over the same axis, i.e. X. This axis is the common one to the two planes used to define the 3D line.*

Theorem 2. *The set of all the lines parallel to a given naive 3D line defined by Figueiredo-Reveillès tiles the space.*

The proof can be deduced easily from the definition of a naive 3D line.

Proof. Given a discrete naive 3D line L, it is characterized by its two projections:

- (On plane P_1) $L_1(a_1, b_1, c_1, \omega_1)$
- (On plane P_2) $L_2(a_2, b_2, c_2, \omega_2)$

Each point $P(u, v, t)$ of the space can be projected on P_1 and P_2. The Bresenham 2D line L_1 tiles the plane P_1, by definition. So there exists a unique $k_1 \in \mathrm{Z}$ such that the projection of P in the plane P_1 belongs to L_{k_1}. The same argument holds for L_2 with respect to the plane P_2.

The lines L_{k_1} and L_{k_2} define a *unique* Figueiredo-Reveillès' naive 3D line according to its definition.

4 Discrete Ray-Casting using Figueiredo-Reveillès' Naive 3D Lines

In this section, we will examine how to develop a discrete ray-casting algorithm using the naive 3D lines defined previously and having the important property that *each point of the scene is reached once* (i.e. no voxel is missed and no voxel is reached more than once). This property holds only if we use a parallel projection and the definition of Figueiredo-Reveillès to define the rays, since Theorem 2 ensures that the set of all the lines parallel to a given discrete line tiles the space.

4.1 Sketch of the algorithm

Let C be the center of the parallelepipedic orthothetic discrete scene bounded by $(0,0,0)$ and $(u_{\max}, v_{\max}, t_{\max})$ that we want to visualize.

As we are using parallel projection, the point of view is placed at infinity. To define the viewing direction, we will consider a point O on the line connecting this point of view to C, O being at finite distance of C. The algorithm is divided in two main sections: *the initializations* and the *main loop*.

- The initialization process consists in (1) selecting the horizontal and the vertical planes for the 2D projections of the Figueiredo-Reveillès 3D lines. (2) The initialization process then computes the coefficients of the corresponding 2D Bresenham lines in the above selected planes. (3) The last initialization step computes the 2D boundaries of the projection of the volume in order to restrict the subsequent 3D lines generation to the actual data volume.
- The main loop consists in (1) computing the entry points into the data volume for all the 3D lines, then these lines are (2) incrementally generated until either they reach a none empty voxel or they exit from the data volume.

4.2 Initializations

1. To select the two 2D projection planes P_1, and P_2 corresponding to the 3D discrete line connecting $O(u_0, v_0, t_0)$ to $C(u_c, v_c, t_c)$, we compute: $m = \max(|\Delta_u|, |\Delta_v|, |\Delta_t|)$, where $\Delta_u = u_0 - u_c$,$\Delta_v = v_0 - v_c$,$\Delta_t = t_0 - t_c$. If $m = |\Delta_u|$ then (horizontal plane) $P_1 = Ouv$ and (vertical plane) $P_2 = Out$. If $m = |\Delta_v|$ then (horizontal plane) $P_1 = Ouv$ and (vertical plane) $P_2 = Ovt$. If $m = |\Delta_t|$ then (horizontal plane) $P_1 = Ovt$ and (vertical plane) $P_2 = Out$.
2. Let D_{k_1} (resp. D'_{k_2}), be the 2D projection of the discrete 3D Figueiredo-Reveillès in plane P_1 (resp. P_2). We can suppose, without any loss of generality, that the slope of these 2D lines is in $[0,1]$ (other cases are symmetric). D_{k_1} is defined by: $a = \Delta_v$, $b = -\Delta_u$, $c = \left\lceil \frac{\omega}{2} \right\rceil - (au_0 + bv_0 + k_1\omega)$, $\omega = \max(|a|, |b|)$ (resp. D'_{k_2} is defined by: $a = \Delta_t$, $b = -\Delta_u$, $c = \left\lceil \frac{\omega}{2} \right\rceil - (au_0 + bt_0 + k_2\omega)$, $\omega = \max(|a|, |b|)$).

3. To know which rays must be sent through the data volume, we define the intervals in which k_1 and k_2 are bounded, using the projections of the data volume vertices. These intervals are: $k_1 \in [k_{1\,\min}, k_{1\,\max}]$ and $k_2 \in [k_{2\,\min}, k_{2\,\max}]$ with: $k_{1\,\min} = K_D(0, u_{\max})$, $k_{1\,\max} = K_D(v_{\max}, 0)$, $k_{2\,\min} = K_{D'}(0, u_{\max})$, $k_{2\,\max} = K_{D'}(t_{\max}, 0)$.

4.3 Main Loop

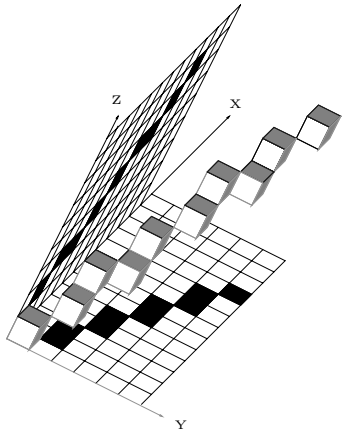

Fig. 1. A Discrete Naive 3D Line

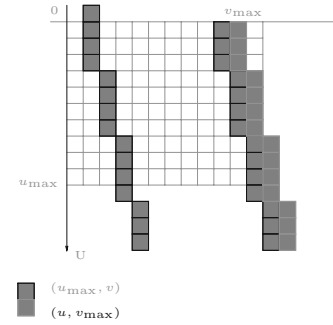

Fig. 2. Intersection

1. For each discrete 3D ray, we compute the intersection with the data volume. As a Figueiredo-Reveillès' line is characterized by two Bresenham lines, and as these two 2D lines are functional over the same axis (Theorem 1), the problem of finding the 3D intersection is reduced to two 2D intersection computations. Furthermore, as we assume that the two 2D lines are functional over the X axis, the entry point to the data volume is $(u, V(D_{k_1}, u), T(D'_{k_2}, u))$ with $u \in [0, u_{\max}]$, $V(D_{k_1}, u) \in [0, v_{\max}]$, $T(D'_{k_2}, u) \in [0, t_{\max}]$, $V(D_{k_1}, u) = -\left[\frac{a_1 u + c_1 - k_1 b_1}{b_1}\right]$, and $T(D'_{k_2}, u) = -\left[\frac{a_2 u + c_2 - k_2 b_2}{b_2}\right]$.
We now consider the problem in the plane P_1, (the similar problem in P_2 is not discussed here). As the current projected line in P_1 lies within the first octant (hypothesis), D_{k_1} can only cut the projection of the 3D data volume across two edges: Edge 1: connecting $(u_{\max}, 0)$ to $(u_{\max}, v_{\max})$ or Edge 2: connecting $(u_{\max}, v_{\max})$ to $(0, v_{\max})$.
In the first case, the target point is $u = u_{\max}$ even if the line has a long run as depicted on Figure 2.
In the second case, we have to find the point $(u, v_{\max})$. If this point does not exist, which may happen since u must satisfy the constraint $u \in [0, u_{\max}[$,

the intersection is void. This means that the corresponding 3D ray does not hit the volume. To compute this intersection point, if any, we calculate the discrete intersection of the line L_1 and the line $v = v_{\max}$ [17]. This intersection is usually composed of more than one pixel, but in our case, we can be sure that they do belong to the same run. The intersection pixel is the first pixel according to the way of generating the discrete line. It is the pixel with the smallest remainder, since the remainder is growing incrementally along a given run. Futhermore this pixel can be found directly using the arithmetical definition of the discrete line.

Using this method we obtain u_1 in the plane P_1 such as $(u_1, V(L_1, u_1)) \in$ box $(0, 0, u_{\max}, v_{\max})$[1] and u_2 in the plane P_2 such as $(u_2, T(L_2, u_2)) \in$ box $(0, 0, u_{\max}, t_{\max})$. Since L_1 and L_2 are functional over the same axis U, the entry point of the ray in the scene is defined by: $(u, V(L_1, u), T(L_2, u))$, where $u = \min(u_1, u_2)$.

2. The Figueiredo-Reveillès' naive 3D line generation algorithm is a straight forward consequence of the generation algorithm a 2D Bresenham Line. Given two points A and B of the space, without any assumption about their positions, Algorithm 2 shows how to generate the Figueiredo-Reveillès' naive 3D discrete segment $[AB]$. The first part of the algorithm determines in which octant B is, when A is considered as the center of the space. The horizontal plane and the vertical plane are chosen. Then two Bresenham algorithms are run: at each step, we consider the next coordinate over X, since the choice of the next coordinate over Y or Z depends on the remainder r_1 and r_2.

4.4 Discrete Z-Buffer

The Ray-Casting method allows to determine the visible points in the scene, but to produce a correct picture with these data, we need to be able to compute the screen to object distance in order to produce an approximation to normal vectors.

Most of well-known methods, such as Z-Buffer shading, need to know for a given point its distance to a fixed point: i.e. the screen or the point of view. As we are using a parallel projection, we can't refer to such a fixed point.

Here again, we are using a discrete approach. We refer to the point O, and we define a naive plane going through O and normal to the discrete 3D line OC. As the set P_k of all the naive planes parallel to a given one tiles the space, we can use k to define the depth of any voxel of the scene according to the plane going through O.

This naive plan is defined by:

$a = u_c - u_0$, $b = v_c - v_0$, $c = t_c - t_0$, $d = \left[\frac{\omega}{2}\right] - (au_0 + bv_0 + ct_0)$, and $\omega = \max(|a|, |b|, |c|)$.

Normal vectors are computed using forward differences in the discrete Z-Buffer, enhancements can be implemented here using [11,16].

[1] box(a,b,c,d) denotes the rectangular 2D orthothetic discrete box of opposite vertices (a,b) and (c,d).

Algorithm 1 Algorithm generating Figueiredo-Reveillès' naive 3D discrete line

Required : $A, B \in \mathbb{Z}^3$
Ensures : procedure PLOT is called for each point of $[AB]$.
  $\Delta[0] \leftarrow B[0] - A[0]$
  $\Delta[1] \leftarrow B[1] - A[1]$
  $\Delta[2] \leftarrow B[2] - A[2]$
  if $\Delta[0] \geq \Delta[1]$ et $\Delta[0] \geq \Delta[2]$ **then**
    $X \leftarrow 0$; $Y \leftarrow 1$; $Z \leftarrow 2$
  else if $\Delta[1] \geq \Delta[0]$ et $\Delta[1] \geq \Delta[2]$ **then**
    $X \leftarrow 1$; $Y \leftarrow 0$; $Z \leftarrow 2$
  else if $\Delta[2] \geq \Delta[0]$ et $\Delta[2] \geq \Delta[1]$ **then**
    $X \leftarrow 2$; $Y \leftarrow 1$; $Z \leftarrow 0$
  end if
  $pt \leftarrow A$
  $i \leftarrow 0$
  $r_1 \leftarrow \left\lfloor \frac{|\Delta[X]|}{2} \right\rfloor$; $r_2 \leftarrow \left\lfloor \frac{|\Delta[X]|}{2} \right\rfloor$
  repeat
    PLOT(pt)
    $i \leftarrow i + 1$
    $pt[X] \leftarrow pt[X] + \text{sign}\,(\Delta[X])$
    $r_1 \leftarrow r_1 + |\Delta[Y]|$
    if $r_1 \geq |\Delta[X]|$ **then**
      $r_1 \leftarrow r_1 - |\Delta[X]|$
      $pt[Y] \leftarrow pt[Y] + \text{sign}\,(\Delta[X])$
    end if
    $r_2 \leftarrow r_2 + |\Delta[Z]|$
    if $r_2 \geq |\Delta[X]|$ **then**
      $r_2 \leftarrow r_2 - |\Delta[X]|$
      $pt[Z] \leftarrow pt[Z] + \text{sign}\,(\Delta[X])$
    end if
  until $i > |\Delta[X]|$

4.5 Geometric Correction

The intermediate screen resulting from the ray-casting algorithm needs to be rescaled to fit the final rectangular screen. Although this geometric correction can easily be performed using conventional methods [10], we have implemented a discrete affine mapping method. This approach is based on the fact that the Bresenham 2D lines tile the plane, so we project runs of adequate lines over single pixels, in order to rescale the whole intermediate screen.

5 Results

Visual results are illustrated Table 3, and 4. Runtime comparisons are given between our program and "Bob" from Graphics and Visualization Lab of the

Army High Performance Computing Research Center, Aberdeen, MD (USA), a freeware conventional real number based visualization tool[2], and "volume" a home made discrete visualization tool based on the scan of the hole discrete 3D data space volume.

	$128^3 voxels$	$150^3 voxels$
bob	n/a	1 min 17 sec
volume	0.49 sec	0.55 sec
Discrete Ray Casting	0.15 sec	0.21 sec

6 Concluding Remarks

We presented a method for volume viewing exclusively based on the use of integer arithmetic. The originality of our algorithm is the use of Figueiredo-Reveillès' 3D discrete lines to model the rays, which ensures that each voxel of the scene is reached only once, since these rays tile the data volume. The front to back algorithm used ensures that only the voxels in front of the visible object are scanned. All this leads to interesting overall runtimes. This program can, in addition, be interfaced with any usual shading algorithm.

Extensions of this work can be integrated into the rendering part of a discrete modeler. The discrete rays can be used in any other field, for example in radio wave propagation simulations.

7 Acknowledgements

The authors express their thanks to Professor Jean Françon for his scientific support and encouragement. Special thanks to the colleagues of LSIIT - Strasbourg for creative and fruitful discussions.

[2] `http://www.arc.umn.edu/`

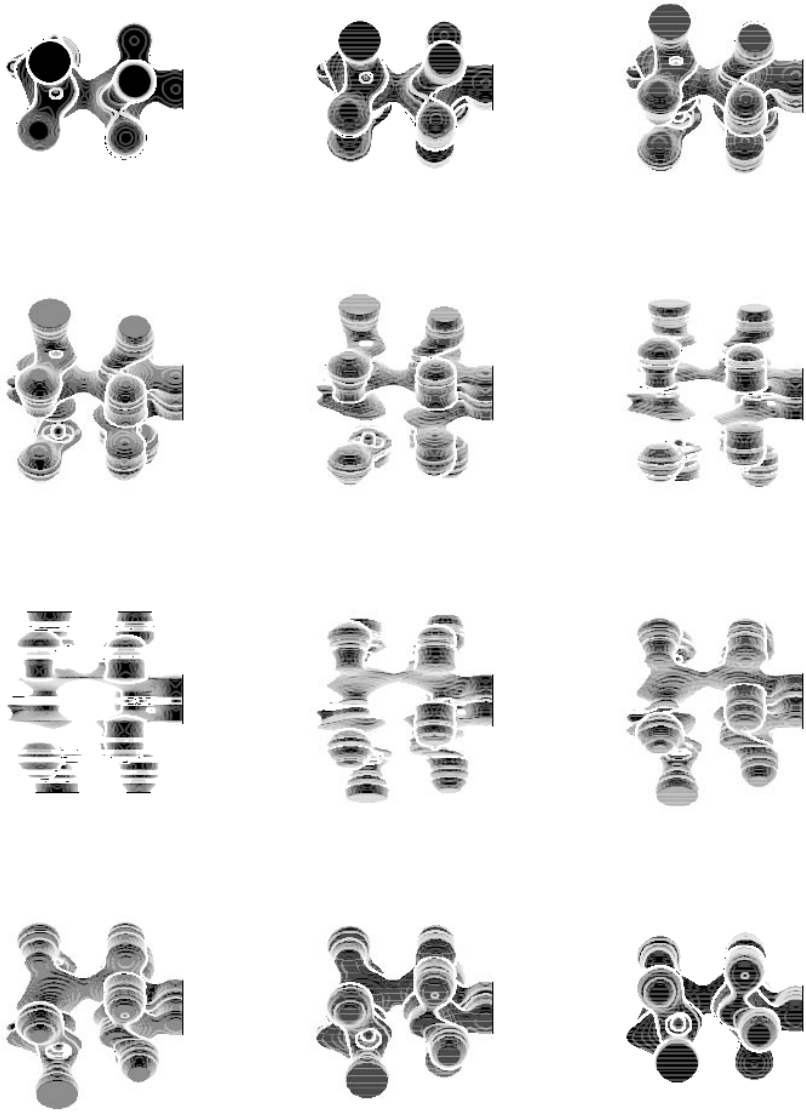

Fig. 3. Twelve views of a Molecule

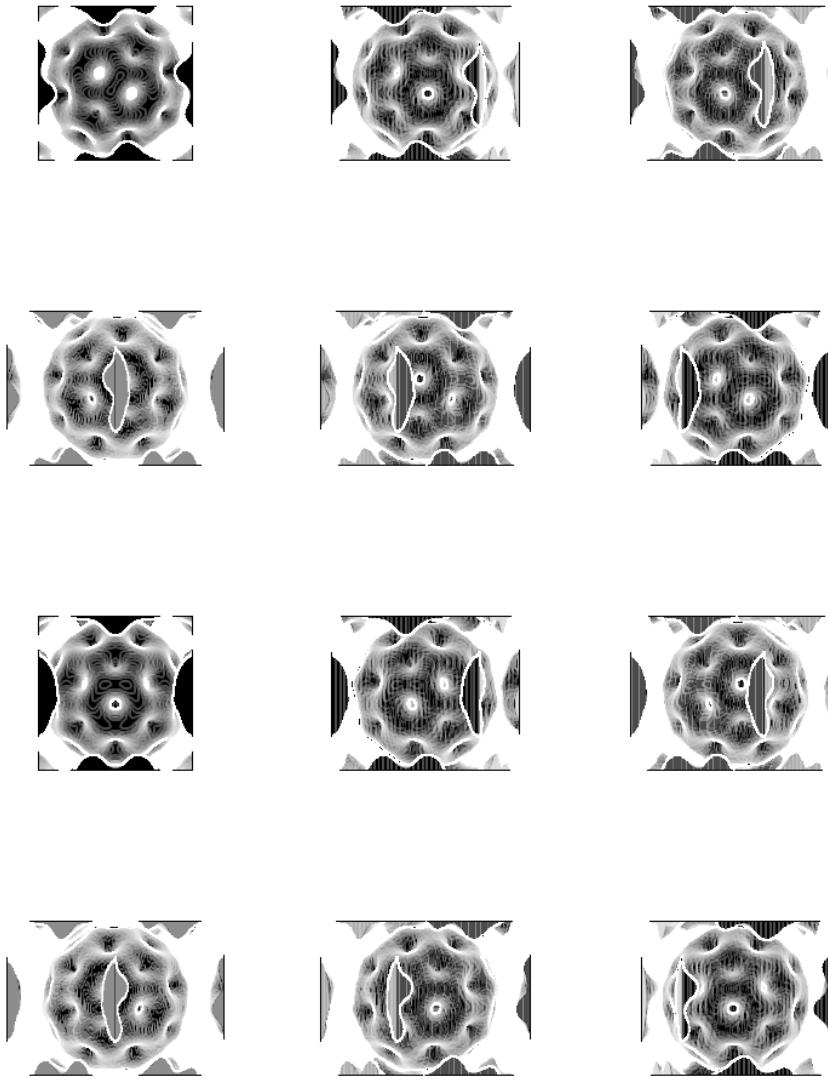

Fig. 4. Twelve views of a C60-Molecule

References

1. E. Andres, *Cercles discrets et rotations discrètes* (in French), Thèse de doctorat, Université Louis Pasteur, Strasbourg, December 1992.
2. E. Andres, R. Acharya, C. Sibata, *Discrete Analytical Hyperplane*, Graphical Models and Image Processing, vol. 59, n.5, Sept. 1997, pp. 302–309.
3. R. Barneva, V. Brimkov, P. Nehlig, *Thin Discrete Triangular Meshes*, to be published in TCS, preliminary version available as research report # 98/01, LSIIT, Université Louis Pasteur, Pôle API, bvd S. Brand, 67400 Illkirch FRANCE
4. J.E. Bresenham, *Algorithm for computer control of a digital plotter*, ACM. Transaction on Graphics, Vol. 4, No 1, 1965, pp. 25–30.
5. J. Delfosse, W.T. Hewitt, M. Mériaux, *An investigation of discrete Ray-Tracing*, 4th DGCI, 1994, Grenoble, pp. 65–76.
6. J. Delfosse, *Le Rendu Discret : Voxélisation, Structures Spaciales et Améliorations* (in French), Thèse de doctorat, Université de Poitiers, Décembre 1996.
7. O. Figueiredo, J.-P. Reveillès, *A Contribution to 3D Digital Lines,* DGCI, 5th Int. Workshop, Clermont-Ferrand, France, September 25-27, 1995, pp. 187–198.
8. J. Françon, *Sur la topologie d'un plan arithmétique*, Theoretical Computer Science, 156, 1996, pp. 159–176.
9. D. Gordon, R. A. Reynolds, *Image Space Shading of 3-dimensional Objects*, Computer Vision, Graphics, and Image Processing, 29 (1985), pp. 361–376.
10. P. S. Heckbert, *Survey of Texture Mapping* IEEE Computer Graphics & Applications, 6, 11 (November 1986), pp. 56–67.
11. A. Kaufman, D. Cohen, R. Yagel, *Normal estimation in 3D discrete space*, The Visual Computer, 1992, 8 pp. 278–291.
12. A. Kaufman, D. Cohen, R. Yagel, *Volume graphics*, IEEE Computer, Vol. 27, No 7, July 1993, pp. 51–64.
13. P. Nehlig, *Applications affines discrètes et antialiassage*, Thèse de doctorat, Université Louis Pasteur, Strasbourg, November 1992.
14. P. Nehlig, C. Montani, *A Discrete Template Based Plane Casting Algorithm For Volume Viewing*, 5th Colloquium: Discrete Geometry for Computer Imagery, Clermont-Ferrand, France, September 25–27, 1995, pp. 71–81.
15. H. Oswald, W. Kropatsch, F. Leberl, *A Perspective Projection with Fast Evaluation of Visibility for Discrete Three-Dimensional Scenes*, Proc. ISMIII '82, International Symposium on Medical and Image Interpretation, IEEE Computer Society Press, Silver Spring, MD (1982), pp. 464–468.
16. L. Papier, J. Françon, *Evaluation de la normale au bord d'un objet discret 3D*, Revue Internationale de CFAO et d'infographie, Ed. Hermes, december 1998.
17. J.-P. Reveillès, *Géométrie discrète, calcul en nombres entiers et algorithmique* (in French), thèse d'état, Université Louis Pasteur, Strasbourg, December 1991.
18. H. K. Tuy, L. T. Tuy, *Direct 2-D Display of 3-D Objects*, IEEE Computer Graphics Appl. 4, 10 (1984), pp. 29–33.
19. R. Yagel, A. Kaufman, *Template-Based Volume Viewing*, Computer Graphics Forum, Eurographic 92 Conference issue (Cambridge, UK) 11, 3 (September 1992), pp. 153–157