Fractal rendering of arbitrary Catmull-Clark surfaces

Krzysztof Gdawiec[1], **Wiesław Kotarski**[2], **Agnieszka Lisowska**[2]

[1] Institute of Mathematics, University of Silesia
kgdawiec@ux2.math.us.edu.pl
[2] Institute of Computer Science, University of Silesia
{kotarski, alisow}@ux2.math.us.edu.pl

Abstract. *In the paper we deal with the fractal rendering of arbitrary Catmull-Clark surfaces. To obtain Iterated Function System (IFS) needed for surface generation we use some facts about approximation of Catmull-Clark surface and fractal description of bicubic patches. First we approximate the given Catmull-Clark surface with bicubic Bézier patches and then for each patch we find corresponding IFS. In this way we obtain fractal description of the surface and therefore we can generate it fractally. Further, some examples of Catmull-Clark surfaces rendered fractally are also presented.*

1. Introduction. The notion of fractal was introduced by Benoit Mandelbrot in the 1970's. But fractals existed considerably earlier. They were perceived as exceptional objects, mathematical monsters. With time they became a very important tool in many disciplines and found very wide practical applications, e.g. image compression, generating of shore lines, mountains, clouds, pattern recognition, image processing or in medicine and economy.

The main property of fractals – self-similarity is the crucial feature used in finding of fractal parameters needed in modelling process. Self-similarity means that the whole object is similar to its parts. For commonly known fractals, such as Koch's Curve or Sierpinski Gasket, self-similarity property is evidently seen. But smooth curves or surfaces usually we do not treat as fractals although in reality they are, which was shown in [9]. Based on this fact in [4] [6] we have shown how to fractally model arbitrary 2D contour and in [4] [5] we have shown fractal rendering of 3D shapes which are sum of Bézier patches. In [3] we used fractal interpolation to obtain the fractal description of a 2D contour.

In this paper we present a method for obtaining a fractal description of any given Catmull-Clark surface which is later used for rendering using the well known fractal generation algorithms. When we have a Catmull-Clark surface first we approximate it with bicubic Bézier patches using the result from [7]. Next, using the result from [5] for each of the Bézier patches we find corresponding IFS obtaining the so-called Partitioned Iterated Function System (PIFS). Having fractal description of the Catmull-Clark surface to render it we use one of the fractal generation methods [8], e.g. deterministic method or chaos game.

In section 2 we briefly present information about Catmull-Clark surfaces. In next section we describe how to approximate given Catmull-Clark surface with bicubic Bézier patches. Because the notion of fractal has many definitions in section 4 we introduce the definition which we will use in this paper. In section 5 we give the formulas for the IFS of bicubic Bézier patch. Later we present a pipeline used to fractal rendering of any Catmull-Clark surface. In section 7 we present some examples of rendering of Catmull-Clark surfaces. Finally, we present some conclusions and tell about the future work.

2. Catmull-Clark surface. Subdivision surfaces have become a standard modelling primitive in 3D games and computer generated motion pictures, e.g. Geri's Game, A Bug's Life, Toy Story 2. Companies, such as Pixar and Alias/Wavefront, have made subdivision surfaces the basic building block for much of their computer graphics software [10]. To create a subdivision surface we

construct a coarse polygon mesh (control mesh) that approximates the shape of the desired surface. When we have such control mesh, a subdivision algorithm recursively refines this mesh producing a sequence of finer meshes which converge to a smooth limit surface. An example of a subdivision surface is a Catmull-Clark surface. It is based on the tensor product of bicubic splines. The scheme produces surfaces that are C^2 everywhere except at extraordinary vertices, where they are C^1 [11].

Figure 1 presents masks used in the iteration process for the Catmull-Clark surfaces. The top-left mask is used for face vertex, top-right for edge vertex, the mask in the middle is used for interior vertex and the masks in the bottom are used for crease and boundary vertices. The coefficients in these masks must sum up to 1, so we need to do a normalization (i.e. divide each of the coefficients by the sum of all the coefficients in the mask). Figure 2 presents an example of a Catmull-Clark surface creation process. We see the initial mesh and the first three iterations.

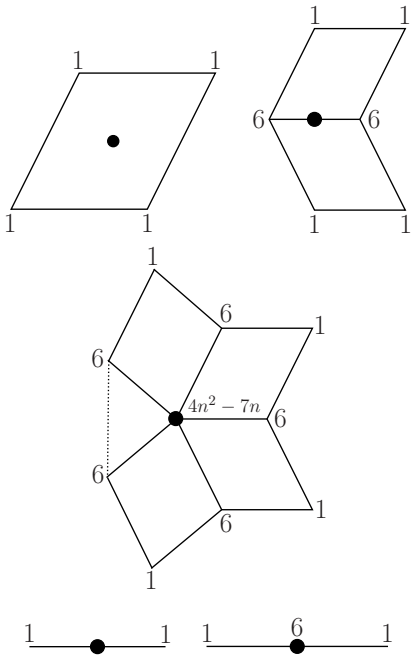

Fig. 1. Masks for the Catmull-Clark surfaces.

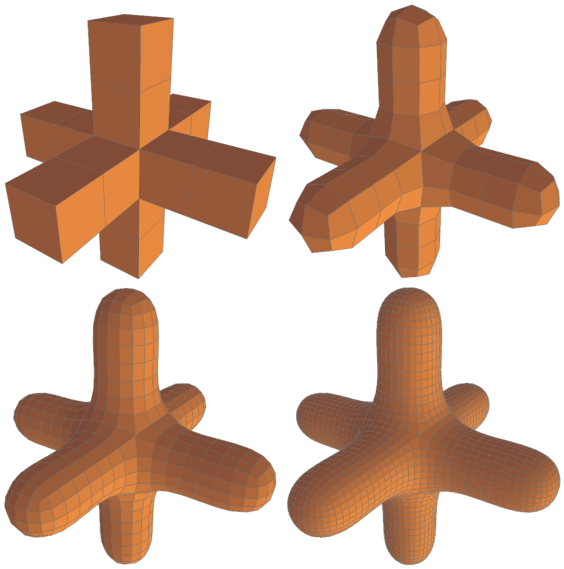

Fig. 2. Initial mesh and 1th, 2nd, 3rd iteration in creation process of the Catmull-Clark surface.

3. Approximation of Catmull-Clark surface with bicubic Bézier patches. In [7] we find an algorithm which allows us to approximate any Catmull-Clark surface with bicubic Bézier patches. The number of approximating patches is equal to the number of faces in control mesh of the surface.

Figure 3 presents mesh of control points for the bicubic Bézier patch and the enumeration of these points. In this mesh we distinguish three types of points:

- interior points (P_{11}, P_{21}, P_{12}, P_{22}),
- edge points (P_{10}, P_{20}, P_{01}, P_{02}, P_{31}, P_{32}, P_{13}, P_{23}),
- corner points (P_{00}, P_{30}, P_{02}, P_{33}).

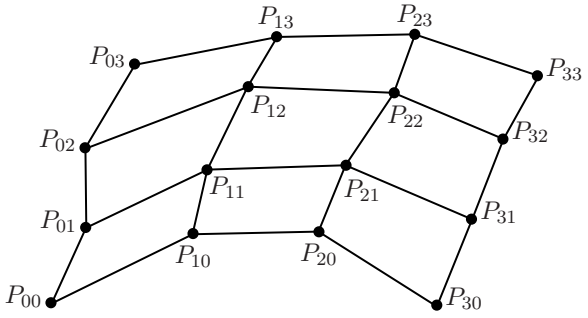

Fig. 3. Control mesh for the bicubic Bézier patch.

To create an approximation of a given Catmull-Clark surface we create one bicubic

Bézier patch for each face of the control mesh. Figure 4 presents masks used to create bicubic Bézier patch in the case of closed Catmull-Clark surface. Like in the masks for Catmull-Clark surface we need to normalize the coefficients. The mask on the top-left is used to create interior points of the patch, the top-right mask is used to create edge points and the bottom mask is used for the corner points. In all cases n is the valance of the vertex.

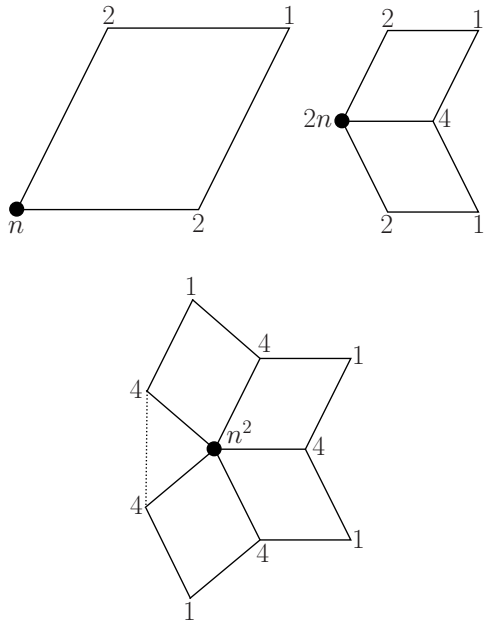

Fig. 4. Masks used in approximation of closed Catmull-Clark surfaces.

In the case of meshes with boundaries (open meshes) we change the masks along the boundary. Figure 5 presents the masks for edge and corner points along the boundary in open mesh. On the top-left there is the mask for edge control point, the top-right mask is for corner point and on the bottom there is the mask for corner point belonging to only one quad. Figure 6 presents the masks for the interior points adjacent to a boundary vertex. On the left there is the mask for interior point adjacent to a boundary vertex belonging to more than one quad and on the right there is the mask for the interior point adjacent to a boundary vertex belonging to only one quad. k is the number of quads containing the boundary vertex and the boundary is indicated by the bolded edges.

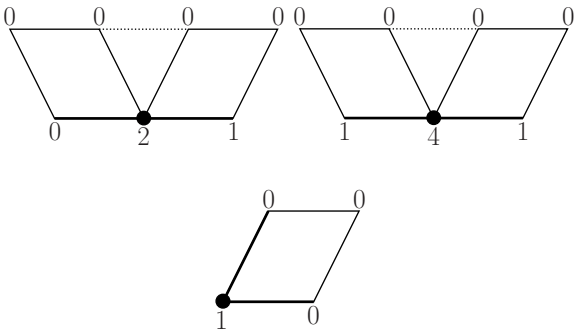

Fig. 5. Masks for edge and corner points along the boundary in open mesh.

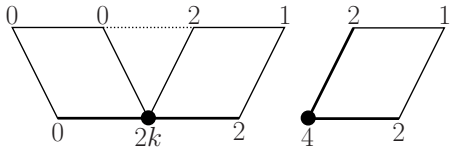

Fig. 6. Masks in open mesh for the interior points adjacent to a boundary vertex.

4. Fractals. In the literature we can find many different non-equivalent definitions of fractal, e.g. an invariant measure, an attractor [1] or the set for which Haussdorf dimension is greater than the topological one [2]. For our purposes the best definition of fractal is an attractor. To give this definition we need to define the notion of an Iterated Function System (IFS).

We say that the set $W = \{w_1, \ldots, w_N\}$, where $w_i : \mathbb{R}^n \to \mathbb{R}^n$ is a contraction mapping for $i = 1, \ldots, N$, of the following form:

$$w_i(x) = A_i \cdot x + t_i, \tag{1}$$

where $A_i \in \mathbb{R}^n_n$ and $t_i \in \mathbb{R}^n$ for $i = 1, \ldots, N$ is an *Iterated Function System*.

An *attractor* of IFS $W = \{w_1, \ldots, w_N\}$ is the limit $\lim_{k\to\infty} W^k(A)$, where A is any nonempty, compact subset of $\mathbb{R}^n$, mapping W is defined as follows:

$$W(A) = \bigcup_{i=1}^{N} \{w_i(a) : a \in A\} \tag{2}$$

and

$$\begin{cases} W^0(A) = A, \\ W^k(A) = W(W^{k-1}(A)) \text{ for } k \geq 1. \end{cases} \tag{3}$$

The simplest way to generate an attractor of a given IFS $W = \{w_1, \ldots, w_N\}$ is to choose any nonempty, compact subset $A \subset \mathbb{R}^n$ and evaluate iterations $W^k(A)$ for $k > 1$. After several iterations we obtain a good approximation of the attractor.

5. IFS for bicubic Bézier patch. In this section we introduce the formula that defines an IFS for bicubic Bézier patch [4] [9]. Figure 3 presents a control mesh for the bicubic Bézier patch. It consists of 16 control points. We divide this mesh into 4 smaller similar meshes (overlapping each other). The enumeration of the points in the smaller meshes stays the same as in the original mesh. Now the four subdivision matrices S_1, S_2, S_3, S_4 defining these meshes are given by the formulas:

$$\begin{aligned} S_1 = L \otimes L, S_2 = L \otimes R, \\ S_3 = R \otimes L, S_4 = R \otimes R, \end{aligned} \tag{4}$$

where

$$\begin{aligned} L &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1/4 & 0 \\ 1/8 & 3/8 & 3/8 & 1/8 \end{bmatrix}, \\ R &= \begin{bmatrix} 1/8 & 3/8 & 3/8 & 1/8 \\ 0 & 1/4 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5}$$

and $\otimes$ denotes the tensor product.

When we obtain these subdivision matrices using the results from [9] we can give the formulas which define the mappings belonging to the IFS for bicubic Bézier patch given by 16 control points. These formulas are

$$w_i = P^{-1} \cdot S_i \cdot P \tag{6}$$

for $i = 1, 2, 3, 4$, where P is the matrix of control points in which the first three columns are the coordinates of the control points, then one column of ones and the rest of the matrix is filled out with random values in such a way that the matrix is non-singular.

Figure 7 presents a fractal rendering example of a bicubic Bézier patch with deterministic method starting from a triangle.

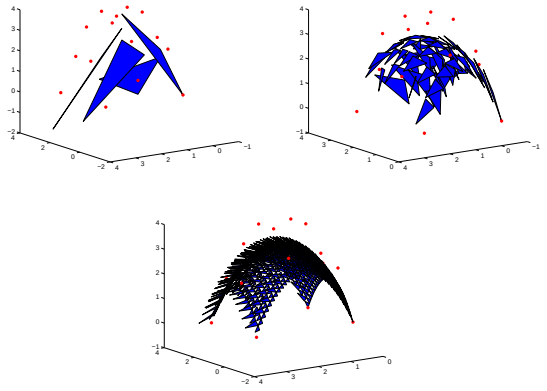

Fig. 7. Fractal rendering of a bicubic Bézier patch – the deterministic method starting from a triangle, iterations: 1, 3, 5.

6. Fractal rendering of arbitrary Catmull-Clark surface. Figure 8 presents the pipeline used to fractal rendering of arbitrary Catmull-Clark surface. First, we approximate it with bicubic Bézier patches using the results from [7] introduced in section 3. Next, for each of the bicubic Bézier patch we find a corresponding IFS. In this way we obtain a set of IFS's and we call it Partitioned Iterated Function System (PIFS). The PIFS is our fractal description of the Catmull-Clark surface. When we obtain such fractal description then we can render the attractor using the well known algorithms, e.g. deterministic method or chaos game [8].

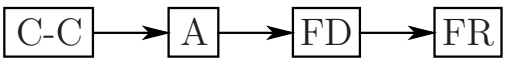

Fig. 8. Pipeline (C-C – Catmull-Clark surface, A – approximation with bicubic patches, FD – fractal description with PIFS, FR – fractal rendering).

7. Examples. In this section we present some examples of Catmull-Clark surfaces rendered fractally. The first example is a shape representing something like letter T. The control mesh consists of 14 faces so there are 14 IFS's representing the shape. Figure 9 presents the control mesh for the Catmull-Clark surface with approximation of it with bicubic Bézier patches and the same surface rendered fractally with deterministic algorithm starting from a point.

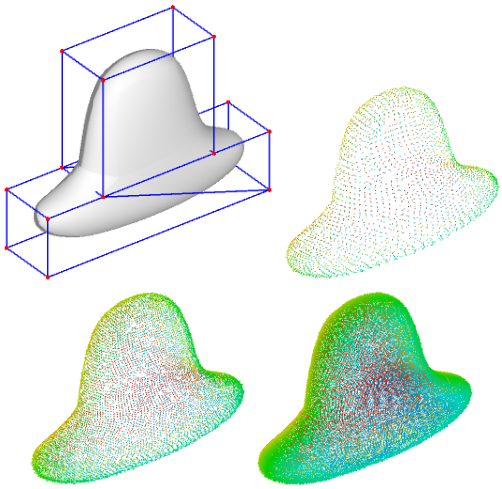

Fig. 9. Letter T – control mesh with approximation and iterations: 4, 5, 6.

Next example presents a ring. It consist of two Catmull-Clark surfaces – the first representing a ringlet (16 faces) and the second one representing a jewel (6 faces). The PIFS consists of 22 IFS's. Figure 10 presents the control mesh of the ring with approximation with bicubic Bézier patches and the 4th, 5th and 6th iteration of the deterministic algorithm starting from a point.

8. Conclusions. In the paper we have presented the algorithm that allows us to find a fractal description of an arbitrary Catmull-Clark surface and to render it fractally by using well-known algorithms. The results presented in the paper can have not only the cognitive meaning but also the practical one. That is why that progressiveness property and resolution independence rendering of graphical objects are highly desired during

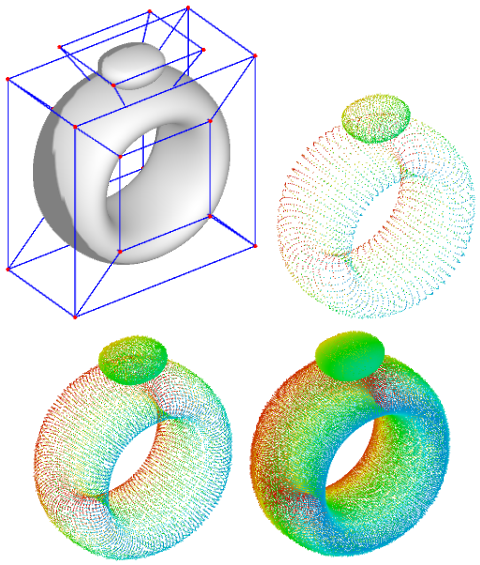

Fig. 10. Ring – control mesh with approximation and iterations: 4, 5, 6.

transmission of the graphical information through a net.

In our further work we will try to create an automatic pipeline for rendering of any given 3D shape which will be similiar to the pipeline for 2D shapes developed by us in [6] and [3].

References.

[1] M. Barnsley, *Fractals Everywhere*, Academic Press, Boston, 1988.

[2] K. Falconer, *Fractal Geometry: Mathematical Foundations and Applications*, John Wiley & Sons, 2003.

[3] K. Gdawiec, *Fractal Interpolation in Modeling of 2D Contours*, International Journal of Pure and Applied Mathematics, vol. 50, no. 3, pp. 421-430, 2009.

[4] W. Kotarski, *Fraktalne modelowanie kształtu*, Wydawnictwo EXIT, Warszawa, 2008.

[5] W. Kotarski, A. Lisowska, *Fractal Rendering of 3D Shapes*, Proceedings of IECON '06 Conference, Paris, (November 7-11, 2006), pp. 3391-3396.

[6] W. Kotarski, A. Lisowska, *On Bézier-fractal Modeling of 2D Shapes*, International Journal of Pure and Applied Mathematics, vol. 24, no. 1, pp. 123-134, 2005.

[7] C. Loop, S. Schaeffer, *Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches*, ACM Transactions on Graphics, vol. 27, no. 1, pp. 8:1-8:11, 2008.

[8] S. Nikiel, *Iterated Function Systems for Real-Time Image Synthesis*, Springer-Verlag, London, 2007.

[9] S. Schaeffer, D. Levin, R. Goldman, *Subdivision Schemes and Attractors*, Eurographics Symposium on Geometry Processing, pp. 171-180, 2005.

[10] J. Warren, H. Weimer, *Subdivision Methods for Geometric Design: A Constructive Approach*, Morgan Kaufmann Publishers, 2002.

[11] D. Zorin, P. Schröder, *Subdivision for Modeling and Animation*, SIGGRAPH 2000 Course Notes.