Barycentric Combinations Based Subdivision Shaders

Lucas Morlet, Marc Neveu, Sandrine Lanquetin, and Christian Gentil
Laboratoire Electronique, Informatique et Image
UFR Sciences et Techniques, allée Alain Savary, 21000 Dijon, France
{lucas.morlet, marc.neveu, sandrine.lanquetin, christian.gentil}@u-bourgogne.fr

ABSTRACT

We present a new representation of uniform subdivision surfaces based on Iterated Functions Systems formalism. Main advantages of this new representation are the formalization of topological subdivision, multiscale representation of limit surface, separation of iterative space where the attractor is computed once for all and modeling space where the attractor is projected many times. An important consequence of this approach is that all uniform subdivision schemes are handled in the same way whatever there are primal or dual, approximating or interpolating.

Subdivision surfaces are no longer viewed as a set of rules but as a list of barycentric combinations to apply on neighborhoods of the coarse mesh. These combinations are representative subsets of the attractor which is deduced from a Controlled Iterated Functions System automaton. From this new point of view we present in this paper a straightforward implementation to directly compute a tessellation of the subdivision surface from a control mesh. This implementation takes full advantage of Graphics Processing Units high capability of computation and Tessellation Stage of OpenGL/GLSL rendering pipeline to generate on the fly a tessellation of the limit surface with a chosen Level of Details.

Keywords

Shape Modeling, Subdivision Surfaces, Iterated Functions Systems, Tessellation Shaders

1 INTRODUCTION

With their overpowered capability of computation, Graphics Processing Units (GPUs) became an unavoidable tool for parallel computing in the last ten years. Whenever a big amount of independent computations are required, an implementation GPU-based must be prioritized over Central Processing Unit (CPU) computations. During the graphics pipeline, several usual steps are highly parallelizable : the vertices positioning and the fragment colorization and blending.

Unfortunately, GPU dedicated memory is limited and transfer times may be longer than computation itself so transmitted information should be as compact as possible. A usual solution is to compress the information before sending them to the GPU and decompress it on the fly during the rendering. This kind of solution reduces occupied memory space and information transfer time but computations are added onto the GPU ; a compromise must be found to maximize the efficiency.

The main scope of this article is on the fly generation of geometry for subdivision surfaces. From a given control mesh, a tessellation of the limit surface is directly computed for a chosen Level of Details (LoD) by applying precomputed barycentric combinations on each patch of the coarse mesh. Patches are defined as in [Sta98] and combinations are deduced from a Controlled Iterated Function System (CIFS) automaton. Indeed barycentric combinations are a representative subset of the attractor of the CIFS which is unique in iterative space but can be projected multiple times in modeling space.

Usually, subdivision surfaces are computed with a set of subdivision rules, which are different for each scheme, applied iteratively on a control mesh. Thanks to this new representation, steps which are blended in these rules are separated : first the iterative process represents by the generation of barycentric combinations with a CIFS automaton, then the projection in modeling space which is the application of combinations on a patch. This separation enables to implement all uniform subdivision schemes in the same way only by changing the input list of precomputed barycentric combinations and the connectivity of input patches.

Another interest of our method is that a vertex position does not depend on the level of tessellation: whatever the chosen LoD, the vertex always belongs to the limit surface. This property is very useful in the CAGD context where usual tools work on limit surfaces rather than meshes. Moreover, isogeometric analysis using subdivision surfaces is more and more often integrated in CAD systems [PXXZ16] [BHU10]. This analysis relies on a compatible representation for geometric modeling and finite element simulation. A general representation of meshes and surfaces, with different LODs is highly recommended to handle isogeometric problems. Our model is quite well adapted for this purpose.

2 RELATED WORKS

Many subdivision schemes have been proposed in the last forty years as [CC78], [DS78], or [Loo87] for instance. From the beginning, authors have paid great attention to the limit surface. Reif [Rei95] proposed a general method to study convergence to the limit surface. Halstead [HKD93] and Stam [Sta98] gave methods to directly compute points on the limit surface. To cover the field of NURBS, extensions have been proposed to cope for non uniform cubic schemes, first by Sederberg [SZSS98], extended by Müller [MRF06] and to high degree surfaces by Cashman [CADS09]. All these schemes present a careful study of the convergence to the limit surface.

Other consideration has been paid on Iterative Function Systems (IFS). In [ZT96], the distinction between the iteration space, where attractors are defined, and the modeling space, where shapes are modeled, enables the generalization of IFS modeling and the definition of attractor projection. Free form fractals and usual free forms curves and surfaces (Bézier, uniform B-splines) can also be defined as an IFS. A link between IFS and subdivision surfaces has been proven very early by Warren and Weimer [WW01] and Shaefer et al [SLG05]. To our knowledge, the interest of the separation between iteration and modeling spaces has not been fully explored for subdivision surfaces.

Indeed, whatever the scheme, different aspects of subdivision surfaces are usually blended in subdivision rules: the connectivity of the control mesh, the subdivision in the parameterization space and the corresponding subdivision in the geometrical space. On the other hand, by using the IFS formalism, we can clearly separate these three aspects and enlarge the possibilities of subdivision surfaces processing. An example can be find in [PGSL14], where Podkorytov builds junctions between two subdivision surfaces, one defined from a primal scheme and the other from a dual scheme.

In video game context, real-time rendering of subdivision surface is an important challenge. Some works [NLMD12][BFK$^+$16] devoted to GPU implementation focus on tessellation. Those works are mainly based on Stam's method [Sta98] with different adaptations. But, they essentially deal with Catmull-Clark subdivision without general formalism. To our knowledge, no proof of a possible extension of their methods to other schemes has been produced.

3 OVERVIEW

To begin, the Iterated Function Systems formalism is briefly presented in Section 4. In particular addresses notion and its influence on equivalence between parametric and barycentric spaces are highlighted as well as projection from barycentric to modeling space.

Then our method is explained step-by-step for the well-known Catmull-Clark subdivision scheme [CC78] in Section 5. First the mesh is cut in several patches. Then barycentric combinations are computed for both regular and irregular cases. To finish, corresponding combinations are applied on each patch to compute the limit surface of the control mesh.

In Section 6, application of our method on several subdivision schemes is presented. Doo-Sabin [DS78], Loop [Loo87], and Simplest [PR97] schemes are handled in the same way as Catmull-Clark scheme with differences only on patches construction and coefficients of barycentric combinations.

A simple and efficient implementation of our method is precisely described in Section 7. It relies on the OpenGL/GLSL rendering pipeline, in particular on the Tessellation Stage which is the core of our implementation. Thanks to our formalism, all subdivision schemes are handled in the same graphics pipeline. This implementation also manages mesh animation, trimming, and dynamic LoD without distinction between subdivision schemes. Some results are presented and comparisons with previous method are discussed.

In Section 8, a problem which appears along with patches size expansion is presented. High-degree or approximating schemes as Butterfly [DLG90] and Quads-interpolating [Kob96] are used as examples to highlight the combinatorial issue.

4 ITERATED FUNCTION SYSTEMS

Iterated Function Systems (IFS) [Hut81][Bar14] are a very efficient tool to represent self-similar objects like fractals, Bézier surfaces, B-spline surfaces... A self-similar object F is defined as an object composed of smaller copies of itself, which can be formalized by : $F = \bigcup_{i=0}^{N-1} T_i(F)$. Each transformation $T_i \in T$ maps F on a subpart of itself. The set of transformations $T = \{T_0 \dots T_{N-1}\}$ is called an IFS. The fixed point F is called the attractor associated to the IFS. According to some conditions, each transformation T_i is contractive for instance, F can be approximated from any initial compact set K_0 : $K_n = \bigcup_{i=0}^{N-1} T_i(K_{n-1})$.

Every point of the attractor corresponds to an infinite sequence of transformations. The sequence of transformation indexes is called the address of the point. The simplest way to obtain points belonging to the attractor is to compute the fixed point P_i of each contractive transformation T_i of the IFS. In the case of affine or linear contractive transformations this can be done by linear algebra. The address of P_i is i^ω (an infinite sequence of i). This means that any point of the attractor, whose address is σi^ω (where σ is a finite word) can be computed in a finite time.

The address function ϕ, mapping an address to its corresponding point, defines a continuous parameterization of the attractor. Given two IFS $\mathscr{I}$ and $\mathscr{I}'$ with the same joining conditions (see [ZT96]), $\phi' \circ \phi^{-1}$ defines a morphism between the two attractors. The first attractor may be used as a parametric space. For a quadrangular surface (like a Bézier Patch), the first attractor could be the unit square defined with an appropriate IFS. The computation of ϕ^{-1} can be achieved with *the escape algorithm* [Bar14].

By choosing the right parameterizations, in a way their addresses have the form σi^{ω}, uniformly spaced and cover the whole parametric space, a tessellation of the attractor can be computed. The maximal authorized length of σ is the level of tessellation of the attractor. The higher it is, the closer from the real attractor the tessellation is. Examples of second tessellation level for two IFS in the parametric space are given in Figure 1.

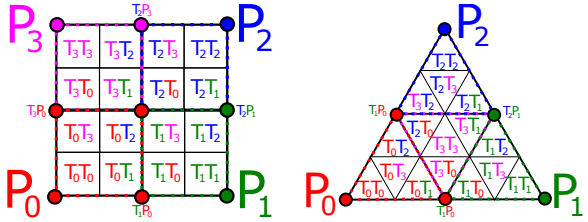


Figure 1: Two examples of IFS in parametric space : on the left, the unit square is cut in four squares ; on the right the "unit" equilateral triangle is cut in four equilateral triangles. Both spaces have four transformations labeled $T_0 \dots T_3$ and associated fixed points $P_0 \dots P_3$

Usually the iterative space, where a tessellation of the attractor is computed, is the modeling space, where the attractor is displayed. In our case, the iterative space is a barycentric space, where every point is a barycentric combination so that the sum of its coordinates is 1. The computed tessellation is then projected in the modeling space by applying the generated combination onto a control mesh. The whole process is summarized in Figure 2.

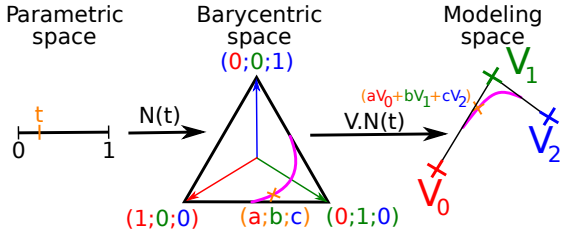


Figure 2: Morphisms between parametric, barycentric, and modeling spaces for a uniform quadratic B-Spline. N is the function that associate a point of the parametric space to the point of barycentric space of same address. V is the projection of a barycentric combination onto a control polygon

The subdivision process consisting in application of the same set of transformations at each level of iteration can be controlled using an automaton describing which transformations can be apply at each state. Such IFS are called Controlled Iterated function Systems (CIFS). The notion of address remains the same and correspond to the set of words accepted by the automaton.

Since subdivision surfaces are iterative models, an IFS and so a CIFS automaton can be created to generate them. For a given list of parametric points, addresses are generated. These addresses are words which are read by the CIFS automaton to compute a set of combinations in a barycentric space. Resulting combinations are then projected onto a control mesh : a tessellation of the attractor, which is the limit surface in the subdivision surface case, is computed.

5 CATMULL-CLARK

For sake of clarity, we first present our method with the well-known Catmull-Clark scheme.

5.1 Patch construction

A *mesh patch* (denoted hereafter *patch* for short) is a set of vertices necessary and sufficient to compute *a piece of the limit surface*. No confusion must be made between these two terms. According to the subdivision scheme, patches have different connectivities. In the Catmull-Clark case, patches are composed of a central face surrounded by a ring of faces [Sta98]. Some examples can be found in Figure 3.

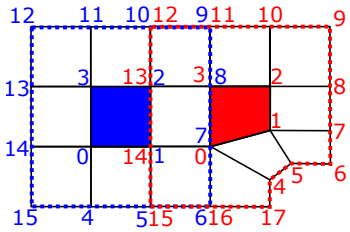


Figure 3: Two examples of Catmull-Clark patches and their indexation. In blue a regular patch (with all the central vertices of valence 4) and in red an irregular patch (with a central vertex of valence 5). Notice that the irregular central vertex, which is unique, is always indexed as 0.

Every central vertex of a regular patch has a regular valency. On the other hand, a patch containing at least one extraordinary vertex (with irregular valency) in its central face is considered as irregular. In this paper, the number of extraordinary vertices per face is limited to a single one to avoid a combinatorial explosion of the number of cases.

When a patch is subdivided, with respect to usual subdivision method, several subpatches are obtained. Each subpatch is also a patch and so can be subdivided at its turn. Every subdivision matrix associates a patch to one of its subpatches. It can be easily deduced from subdivision rules. The set of these transformations is a CIFS

whose the attractor is the piece of the limit surface defined by the patch. This CIFS depends on the valency of the irregular vertex, if it exists. For the sake of simplicity, regular patches are treated first, then generalized to irregular patches.

5.2 Regular patches

In the regular case, a Catmull-Clark subdivision surface is identical to a bicubic uniform B-Spline surface. So regular patches are grids of 4 by 4 vertices. Subpatches are four in number and each one is also a regular patch. The regular patch subdivision is presented in Figure 4.

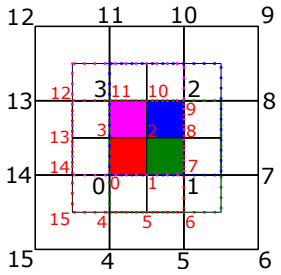


Figure 4: The regular patch subdivision of Catmull-Clark scheme into four regular subpatches. Notice that vertices are indexed in the same way in subpatches as in their parent patch.

Four square subdivision matrices $M_{i\in[0;3]}$ can be created to transform the parent patch in each regular subpatches. The M_0 subdivision matrix is given as an example in Figure 5.

$$M_0 = \left(\begin{array}{cccc|cc|cccc|cccc}
V & E & F & E & E & F & & & & & F & E & F \\
B & B & C & C & C & C & & & & & & & \\
A & A & A & A & & & & & & & & & \\
B & C & C & B & & & & & & & C & C & \\
\hline
B & C & & & B & C & & & & & & C & C \\
A & A & & & A & A & & & & & & & \\
C & B & & & C & B & C & C & & & & & \\
E & V & E & F & F & E & F & E & F & & & & \\
\hline
C & B & B & C & & & & C & C & & & & \\
F & E & V & E & & & & F & E & F & E & F & \\
C & C & B & B & & & & & & C & C & & \\
E & F & E & V & & & & & & F & E & & \\
\hline
C & & & B & & & & & & C & C & B & C \\
A & & & A & & & & & & & A & A & \\
B & & & C & C & & & & & & C & B & C \\
A & & & & A & & & & & & & A & A
\end{array}\right)$$

$$A = \frac{1}{4} \quad B = \frac{3}{8} \quad C = \frac{1}{16}$$

$$V = \frac{9}{16} \quad E = \frac{3}{32} \quad F = \frac{1}{64}$$

Figure 5: The M_0 subdivision matrix for the Catmull-Clark subdivision scheme. Null coefficients are omitted.

These subdivision matrices are stochastic and have a unique eigenvalue equals to 1 and all others in interval $[0;1[$. This implies that the associated transformation is contractive and so apply an infinity of times the transformation on a patch will end in a unique fixed-point. As proven in [HKD93], this fixed-point can be computed directly by using the left eigenvector associated to the eigenvalue 1 as a barycentric combination B_i which associates a patch to a point of the limit surface.

From every given a point $(u;v) = T_aT_b \dots T_yP_z$ in the parametric space, its address is used to compute the corresponding barycentric combination $B_zM_y \dots M_bM_a$. This combination transforms a regular patch into the point of the limit surface of the local parameterization $(u;v)$. Computations of combinations can be expressed as a CIFS automaton whose an example is given in Figure 6.

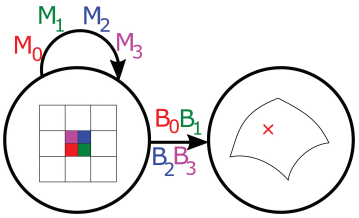


Figure 6: The CIFS automaton for the regular case of Catmull-Clark subdivision scheme.

For a given Level of Details, several parametric points are chosen to represent a discretization of parametric space. These points must be uniformly spaced, cover the whole parametric space, and correspond to a finite address. Then each barycentric combinations associated to these points are computed. Apply all these combinations on the same patch create a tessellation of the limit surface with the chosen Level of Details.

5.3 Irregular patches

A major interest of subdivision schemes is the management of irregular connectivity. For example, bicubic uniform B-Spline surfaces can not be computed on a non-grid mesh. Catmull-Clark scheme, in addition to generate bicubic uniform surfaces in regular case, permits to generate C1-continue surfaces around extraordinary vertices.

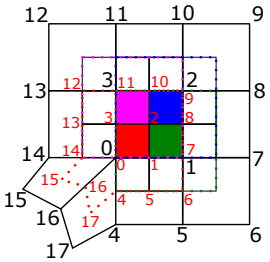
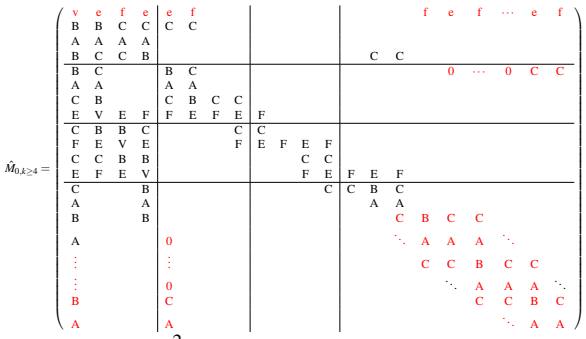


Figure 7: An irregular patch (valence 5) subdivision of Catmull-Clark scheme into four subpatches. Notice that the red subpatch has the same connectivity as its parent patch whereas the other ones become regular patches.

In presence of extraordinary vertices, patch connectivity and subdivision rules are different, so some barycentric combinations should be recomputed. As showed in Figure 7, subpatches connectivities are also different.

As long as the applied transformation is the one centered on the extraordinary vertex, the subpatch keeps the connectivity of its parent. As soon as another transformation is applied, the subpatch becomes regular.

To manage irregular transformations, irregular subdivision matrices $\hat{M}_{i,k}$, which depend on the valency k of the unique extraordinary vertex have to be introduced. $\hat{M}_{i>0,k}$ matrices are not so different from $M_{i>0}$ except there are not square anymore. On the other hand, $\hat{M}_{0,k}$ is square but depends on the valency k of the extraordinary vertex. This matrix is presented in Figure 8.



$$v = \frac{4k^2-7k}{4k^2} \quad e = \frac{6}{4k^2} \quad f = \frac{1}{4k^2}$$

Figure 8: The $\hat{M}_{0,k>4}$ subdivision matrix for the Catmull-Clark subdivision scheme. Black coefficients are the fixed ones and the red depends on the valency of the extraordinary vertex. A, B, C, V, E, and F coefficients are the same as in Figure 5.

Concerning computations of fixed points, $\hat{B}_{i>0,k}$ do not need to be computed because there are equal to $B_i\hat{M}_{i,k}$. Conversely every $\hat{B}_{0,k}$ has to be computed : it is the eigen-vector associated to the eigen-value 1 of $\hat{M}_{0,k}$. Everything is summerized in the automaton of irregular case given in Figure 9.

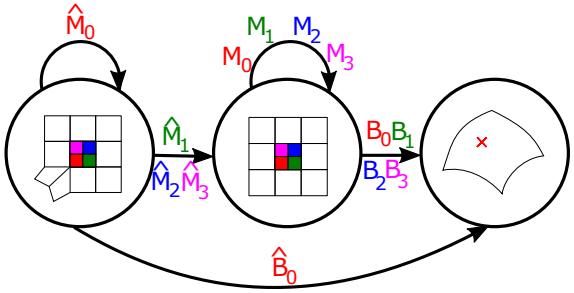


Figure 9: The CIFS automaton associated to irregular cases of Catmull-Clark subdivision scheme.

6 OTHER SUBDIVISION SCHEMES

As said before, our formalism handles any uniform subdivision schemes. In this section, some usual schemes are presented as CIFS automata.

6.1 Loop scheme

As for the Catmull-Clark scheme, the Loop subdivision scheme [Loo87] creates C2-continue surfaces. Patches are created in the same way : a central face and a ring of faces as shown in Figure 10.

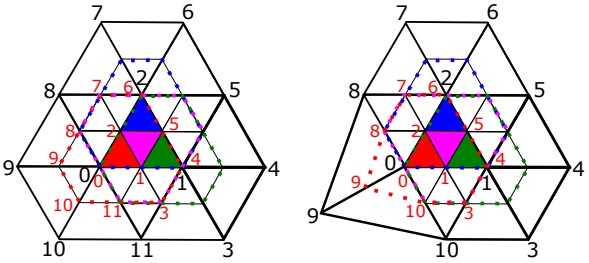


Figure 10: Patches and subpatches of the Loop subdivision scheme for the regular case (valence 6) and an irregular case (valence 5). Notice that the red subpatch is connectively-identical to the parent one and the other three are regular.

The structure of CIFS automaton associated to the Loop scheme is identical to the Catmull-Clark one. There are three states which correspond to irregular patches, the regular patch, and limit surface points and the transitions are labeled exactly in the same way.

Since the subdivision rules of the two schemes are different, the transformation matrices associated to the transitions are not the same in the two automata. Subdivision of the parametric space is also different so addresses are generated differently as shown on the right of Figure 1.

6.2 Doo-Sabin scheme

Unlike Catmull-Clark and Loop subdivision schemes, which are primal, Doo-Sabin [DS78] is a dual subdivision scheme which means that faces are not subdivided anymore but vertices. So corresponding patches are composed of a central vertex and a ring of adjacent faces. The valency of all the vertices is 4 ; irregular patches contain a unique non-quadrilateral face as shown in Figure 11.

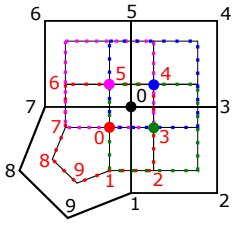


Figure 11: Irregular patch (with a pentagonal face) of the Doo-Sabin subdivision scheme and its four subpatches. Once more, the red subpatch is connectively identical to the parent one and the three others become regular.

At this point it is important to say that Doo-Sabin and Catmull-Clark subdivision schemes are identical from a topological subdivision point of view : it is a quadrangular subdivision with the same kind of connection between subdivided faces. As for the Loop scheme, the CIFS associated to Catmull-Clark and Doo-Sabin subdivision schemes have the same structure (states and transition labels). The only difference to highlight is the difference of transformation matrices.

6.3 Simplest scheme

Introduced by Peters and Reif [PR97] the Simplest subdivision scheme, also called Midedge scheme, inserts a vertex in the middle of each edge and creates new edges between new vertices. Then old vertices and edges are deleted.

As shown in Figure 12, applying the Simplest subdivision scheme twice is connectively-identical with Doo-Sabin but with different coefficients. So the CIFS automaton associated to the Simplest scheme is the same as Doo-Sabin one but coefficients of transformation matrices are different.

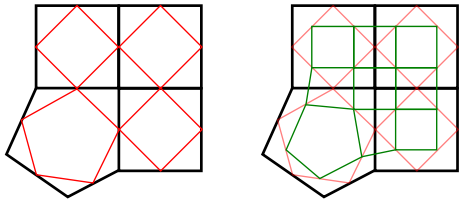


Figure 12: Subpatches of the Simplest subdivision scheme after one (red) and two (green) subdivisions. After two subdivisions, the result is connectively-identical with the Doo-Sabin scheme (see Figure 11).

7 IMPLEMENTATION

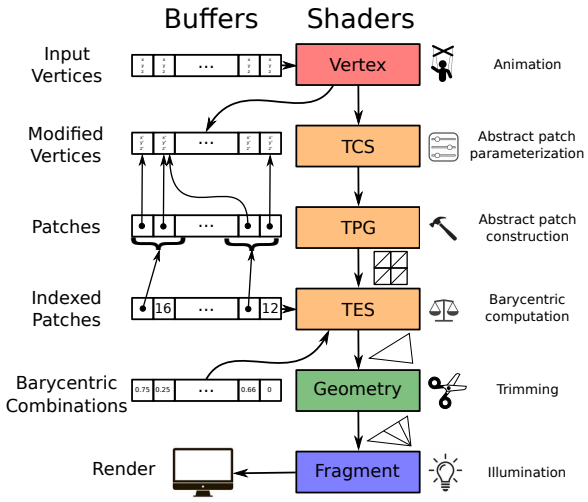


Figure 13: Implementation overview.

In this section, we suggest an implementation of our method in OpenGL/GLSL. Because our implementation needs Shader Storage Buffer Objects, the minimum version required for both core version is 4.3 or 4.0 with ARB_shader_storage_buffer_object extension. An overview is given in Figure 13.

7.1 First steps

For a given maximal Level of Details and a minimal and maximal valence of extraordinary vertices authorized, all combinations are computed for a chosen subdivision scheme. Then meshes are cut in several patches respecting inherent rules of the scheme. These two steps are done once for all and results are written in buffers.

7.2 Buffers

Buffers require to have an array of arbitrary length so they must be Shader Storage Buffer Objects (SSBOs). Referring to OpenGL specifications, SSBOs reads and writes use incoherent memory accesses and guarantee a minimum possibility of memory allocation of 128MB. Most GPU-implementations enable allocating a size up to the limit of GPU memory.

Vertices buffers

There are two vertices buffers : the *Input Vertices Buffer* which contains all vertices of the input control mesh and the *Modified Vertices Buffer* which contains the vertices after modification by the Vertex Shader.

Patches buffers

The GLSL Tessellation Shaders does not handle dynamic patch size. To bypass this constraint, two patches buffers are created. The first one, called *Patches Buffer*, contains all mesh patches written in a row, each patch containing index of its vertices. The second one, the *Indexed Patches Buffer*, indexes patches of *Patches Buffer* by a pointer to the begin and the length of each mesh patch.

Barycentric Combinations Buffer

For a given valence of extraordinary vertex, each barycentric combination contains as many coefficients as vertices in patch. The number of combinations of each patch size depends on the chosen maximal Level of Details. The *Barycentric Combinations Buffer* is a row containing all coefficients of all combinations of each kind of patch. By knowing the maximal Level of Details, the valence of extraordinary vertex, and the parameterization $(u;v)$ of the current limit surface point, the index of associated barycentric combinations can be computed.

7.3 Programmable rendering pipeline

Once all buffers are filled, the usual OpenGL/GLSL rendering pipeline with the succession of shaders is performed.

Vertex Shader

This shader treats one by one all the vertices of *Input Vertices Buffer* and write them into the*Modified Vertices Buffer*. Vertices are ordered in the same way in both

buffers. Vertex Shader can be simply a pass-through shader by copying the *Input Vertices Buffer* into the *Modified Vertices Buffer* but can also be the step where mesh is animated. Animation is discussed in Subsection 7.4.

Tessellation Control Shader (TCS)

This shader configures the Tessellation Primitive Generator (TPG) which creates an *abstract patch* and transfers it to Tessellation Evaluation Shader. An *abstract patch* is a set of parametric points connected with triangles. These triangles are chosen by the TPG in a way to cover all the parametric space. The parametric space can be of two types depending on the *abstract patch* configuration : the unit square or the "unit" equilateral triangle (cf Figure 1). The TCS describes how many points of the *abstract patch* are on each bounding-edge of parametric space and how many interior rings are created but can not choose how to rely them. The more detailed the limit surface is desired, the more important is the number of inserted points. Level of Details can be the same everywhere or adaptive and chosen on the fly for each patch. More information about adaptive LoD is given in Subsection 7.5.

Tessellation Evaluation Shader (TES)

This shader is the core of our implementation. From one side, it reconstitutes the current mesh patch by reading *Indexed Patches Buffer* which points to *Patches Buffer* which contains indexes of vertices in *Modified Vertices Buffer*. From the other side it reads the parameterization of all points of the *abstract patch* and deduced a list of associated barycentric combinations. To finish, it applies every combination on the patch to compute points of limit surface and rely them as describes by the *abstract patch*.

Geometry Shader

This shader takes an OpenGL primitive (a triangle in this case) as input and emits zero or more primitives (triangle-strip). It can be simply a shader which applies the Model-View-Projection matrix on each received triangle but can also be used to trim the limit surface. Subsection 7.6 is dedicated to the trimming of the limit surface by parametric space restriction.

Fragment Shader

As usual, this shader is in charge of coloration and illumination, with a Phong model for example. As proven in [BGN09], normal associate to every point of thelimit surface is the cross-product of the two half-tangents computed for the address of the point.

7.4 Animated meshes

The main idea is instead of animating the subdivide mesh, the coarse mesh is animated and then subdivided. Animation time is greatly reduce but time is spent to generate geometry. In the common case of animation by skeleton with an average of two bones by vertex geometry generation is slightly less two times longer than animation. Resulting surfaces are not the same in both methods : subdivide then animate deforms the limit surface whereas animate then subdivide assures the conservation of limit surface continuity.

7.5 Adaptive Level of Details

An efficient method to avoid the time-consuming display of very detailed distant objects is the LoDs strategy : the closer the object is, the more detailed it is ; the farther it is, the faster it is displayed. In most implementations, several meshes, with different LoDs, are generated and one of them is picked up at display time.

The limit of this approach is that the different meshes have to be predefined. On the contrary, with subdivision surfaces, only the coarse mesh is defined and the LoDs gives the number of subdivision. Iteratively generating geometry on the fly can be quite long with a multipass render. With our method, geometry is not iteratively generated but directly only by selecting the appropriate set of barycentric combinations to apply on the mesh patch.

Another advantage of our method is that an object does not need to be uniformly subdivided (i.e. all faces are subdivided the same number of times). Each face is independently treated from the others, with its own tessellation that can be non-uniform. To avoid cracks, a condition has to be imposed : even faces can be subdivided as wanted, edges are subdivided in the same way on both sides.

A naive method to ensure respect of edge uniform subdivision is to project every face of control mesh in image plane. Each edge is subdivided in function of its screen length and face is subdivided in function of its screen area. The OpenGL's invariance rules insures that multiple projection of the same edge results in the same screen length and so same subdivision for both adjacent sides.

7.6 Trimming

Trimming is a restriction of the parametric space that results in a restriction (holes for instance) on the limit surface. Quality of trimming is strongly dependent on the level of tessellation. To enhance quality, trimmed faces have to be more tessellated. Thanks to dynamic LoDs, untrimmed faces do not need to be as tessellated as trimmed ones.

First, tessellation is traced in parametric space. If a point is outside the restriction it is tagged as valid otherwise not. For every triangle of tessellation, the number of valid vertices is counted. Each case is treated differently. All of them are presented in Figure 14.

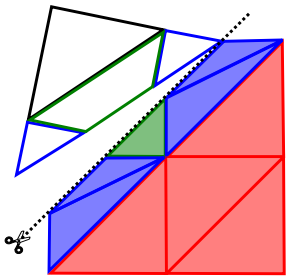


Figure 14: Example of trimming on a parametric space tessellated by 8 original triangles represented by their colored bounds. New triangles are filled if they are displayed, empty if not. Red triangles have three valid vertices so they are displayed as is. Blue triangles have an invalid vertex which becomes a vertex on each adjacent edge and the quadrilateral result is split into two triangles. Green triangle has two invalid vertices so both are recomputed. Black triangle is completely included in the restriction so they are discarded.

The intersections between the triangles edges are new point of parametric space. New points means new addresses and so new barycentric combinations. In order to get valid trimmed boundary points, new combinations have to be computed and applied on the patch.

7.7 Results and performance

In this section, several methods are compared in term of occupied memory space and computation time for the regular case of the Catmull-Clark subdivision scheme. These methods are three in number :

1. LoD meshes : one mesh is defined for each LoD. Only one is picked up and display.
2. Iterative subdivision : only patches of coarse mesh are transfered to the GPU and geometry is generated iteratively with respect to subdivision rules.
3. Ours : as Iterative subdivision, only patches are transfered to the GPU but geometry is generated directly by applying barycentric combinations.

Let n the maximal LoD, V_i the number of vertices and F_i the number of faces/patches of the mesh associated to the LoD of i. k is the number of existing valences. Two tables are given to compare these three methods in term of occupied memory space (Table 1) and computation time (Table 2).

Methods	LoD	It. sub.	Ours
Vertices	V_n	V_0	V_0
Indexed vertices	$4\sum_{i=0}^{n}(4^i)F_0$	$18F_0$	$18F_0$
Combinations	0	0	$k*(2^n+1)^2$

Table 1: Comparisons of occupied memory space between the three methods. A face is defined by 4 pointers to vertices and an average patch by 16 pointers to vertices plus 2 pointers to index the patch.

Methods	LoD	It. sub.	Ours
Animated vertices	V_n	V_0	V_0
Subdivision	0	$\sum_{i=0}^{n-1} F_i$	0
Combinations	0	0	$(2^n+1)^2F_0$

Table 2: Comparisons of computation time between the three methods. Animation of a vertex costs 25 basic operations (times or plus) per bones (matrix-vector product of size 4). A subdivision corresponds to the computation of 1 face-vertex, 4 edge-vertices and 4 vertex-vertices so 267 basics operations. Application of a barycentric combination costs an average of 93 basics operations (application of a 16 coefficients combination onto a patch of three coordinates vertices).

Name	Gaussian	Tetris	Head	Body
Vertices	100	74	4276	13 652
Faces	81	72	4249	13 650
LoD = 0	0.26	0.27	0.79	1.82
LoD = 1	0.26	0.28	1.24	3.27
LoD = 2	0.29	0.32	3.41	10.2
LoD = 3	0.42	0.57	16.4	47.6
LoD = 4	1.06	1.48	62.5	≈200
LoD = 5	2.53	3.58	167	≈500

Table 3: Computation times (in ms) of our method for different meshes and LoD. Tests are performed on a Dell Precision T7600 : Intel Xeon CPU E5-2609 @ 2.40 GHz x8 and a Nvidia Quadro K2000/PCIe/SSE2.

In term of occupied memory space, subdivision methods are obviously better than the LoD method because only the coarse mesh is needed. Because barycentric combinations have to be loaded in the memory, our method required a little more space than iterative method.

In term of computation time, our method is faster than iterative method because application of barycentric combination requires less computation than subdivision. On another side, our implementation is slower than LoD methods because animation is approximatively two times quicker than generation. Even if dynamic Level of Details permits to reduce the gap between the two methods, LoD method is still the fastest one.

Our method has been tested on some meshes : results are presented in Figure 15 and computation times are compared in Table 3.

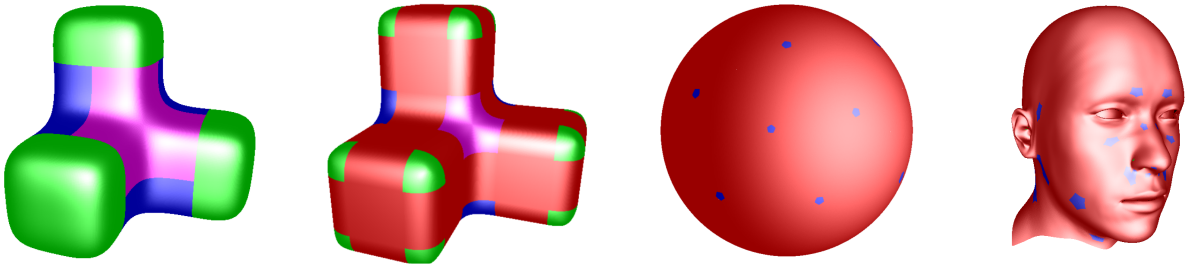


Figure 15: From left to right : Catmull-Clark and Doo-Sabin schemes applied on Tetris meshes ; Loop scheme applied on an Icosphere and Catmull-Clark scheme applied on Human Head mesh enlightened by a Blinn-Phong shader. Red faces correspond to regular patches, green to extraordinary valence of 3, blue of 5 and magenta of 6.

8 LIMITS

Even if our formalism can handle every uniform subdivision scheme, for certain schemes a combinatorial issue appears because of many possibilities of irregularities. In this case, the CIFS automaton still contains a unique regular state but also several irregular states that must be treated separately. Some examples are given in this section.

8.1 High degree schemes

The only condition for building patches is : "at most one extraordinary vertex inside a valid patch" (there is no condition on the exterior ring). The higher the degree of the surface is, the larger the patch is and so more possibilities of irregularities appear as shown in Figure 16.

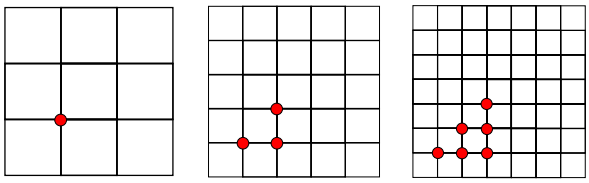


Figure 16: From left to right : subdivision patch for bicubic subdivision (Catmull-Clark), biquintic, and biheptic. Red vertices are the maintained possibilities of extraordinary vertices after reduction by symmetry and rotation. Notice the number of red vertices increases with the degree of surface.

A recurrence appears for odd degree regarding these three examples : the number of possibilities of extraordinary vertices for a surface of degree d is the sum of integer from 1 to $(d-1)/2$.

8.2 Interpolating schemes

Interpolating schemes need larger patches than approximating schemes so they suffer from the same issue as the high-degree schemes. For example Butterfly [DLG90] and Quads-interpolating [Kob96] schemes need a supplementary ring with respect to Catmull-Clark and Loop schemes in order to build a mesh patch. These patches and their subpatches are presented in Figure 17.

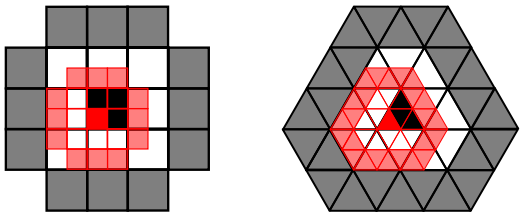


Figure 17: The parent patch and one of its subpatches for the regular case of Quads-interpolating (left) and Butterfly (right) subdivision schemes. Patches are centered on the filled face and the supplementary faces (compared to corresponding approximating schemes) are transparent.

9 CONCLUSION

In this article, subdivision surfaces are presented in the Controlled Iterated Functions Systems formalism. In this formalism, subdivision schemes are not viewed as a set of rules anymore but as a list of barycentric combinations. From this new point of view, all the uniform schemes, whatever they are approximating or interpolating, primal or dual, are handled in the same way. Usual tools as trimming and multi-resolution render are also independent from the chosen scheme.

An implementation based on this formalism is also suggested. This implementation generates directly and on the fly the limit surface from a control mesh faster than usual iterative subdivision surfaces but required a little more memory space. Compared to a usual LoD approach, our method is slower but saves a lot of memory space.

10 REFERENCES

[Bar14] Michael F Barnsley. *Fractals everywhere*. Academic press, 2014.

[BFK+16] Wade Brainerd, Tim Foley, Manuel Kraemer, Henry Moreton, and Matthias Nießner. Efficient gpu rendering of subdivision surfaces using adaptive quadtrees. *ACM Trans. Graph.*, 35(4):113:1–113:12, July 2016.

[BGN09] Hicham Bensoudane, Christian Gentil, and Marc Neveu. Fractional half-tangent of a curve described by Iterated Function Systems. *Journal of Applied Functional Analysis*, 4(2):311–326, April 2009.

[BHU10] Daniel Burkhart, Bernd Hamann, and Georg Umlauf. Iso-geometric Finite Element Analysis Based on Catmull-Clark Subdivision Solids. *Computer Graphics Forum*, 2010.

[CADS09] Thomas J Cashman, Ursula H Augsdörfer, Neil A Dodgson, and Malcolm A Sabin. Nurbs with extraordinary points: high-degree, non-uniform, rational subdivision schemes. In *ACM Transactions on Graphics (TOG)*, volume 28, page 46. ACM, 2009.

[CC78] Edwin Catmull and James Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355, 1978.

[DLG90] Nira Dyn, David Levine, and John A Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM transactions on Graphics (TOG)*, 9(2):160–169, 1990.

[DS78] Daniel Doo and Malcolm Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

[HKD93] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 35–44, New York, NY, USA, 1993. ACM.

[Hut81] John E Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

[Kob96] Leif Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. In *Computer Graphics Forum*, volume 15, pages 409–420. Wiley Online Library, 1996.

[Loo87] Charles Loop. Smooth subdivision surfaces based on triangles. 1987.

[MRF06] Kerstin Müller, Lars Reusche, and Dieter Fellner. Extended subdivision surfaces: Building a bridge between nurbs and catmull-clark surfaces. *ACM Transactions on Graphics (TOG)*, 25(2):268–292, 2006.

[NLMD12] Matthias Nießner, Charles Loop, Mark Meyer, and Tony Derose. Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. *ACM Transactions on Graphics (TOG)*, 31(1):6, 2012.

[PGSL14] Sergey Podkorytov, Christian Gentil, Dmitry Sokolov, and Sandrine Lanquetin. *Joining Primal/Dual Subdivision Surfaces*, pages 403–424. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[PR97] Jörg Peters and Ulrich Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics (TOG)*, 16(4):420–431, 1997.

[PXXZ16] Qing Pan, Guoliang Xu, Gang Xu, and Yongjie Zhang. Isogeometric analysis based on extended catmull-clark subdivision. *Computers & Mathematics with Applications*, 71(1):105 – 119, 2016.

[Rei95] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. volume 12, pages 153–174, 1995.

[SLG05] S. Schaefer, D. Levin, and R. Goldman. Subdivision schemes and attractors. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[Sta98] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 395–404, New York, NY, USA, 1998. ACM.

[SZSS98] Thomas W Sederberg, Jianmin Zheng, David Sewell, and Malcolm Sabin. Non-uniform recursive subdivision surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 387–394. ACM, 1998.

[WW01] Joe Warren and Henrik Weimer. *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2001.

[ZT96] Chems Eddine Zair and Eric Tosan. Fractal modeling using free form techniques. *Computer Graphics Forum*, 15(3), 1996.