

Surfaces de subdivision, NURBS, et fractales :  
vers un modèle géométrique générique  
à base de systèmes itérés et contrôlés de fonctions

par Lucas MORLET  
pour l'obtention du titre de Docteur  
Spécialité doctorale "Informatique"

Soutenue le 6 décembre 2019



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Une histoire de courbes et de surfaces . . . . .	9
1.1.1	Les courbes de Bézier et l'algorithme de Casteljau . . . . .	9
1.1.2	L'ingénierie : les B-Splines et les NURBS . . . . .	10
1.1.3	Le monde virtuel : les surfaces de subdivision . . . . .	10
1.1.4	Les fractales . . . . .	11
1.2	Objectif . . . . .	13
1.3	Organisation générale du manuscrit . . . . .	14
<b>2</b>	<b>Systèmes Itérés (et Contrôlés) de Fonctions</b>	<b>17</b>
2.1	Fractales . . . . .	17
2.1.1	Ensembles de Julia, de Fatou, et de Mandelbrot . . . . .	17
2.1.2	Dimensions fractales . . . . .	18
2.1.3	Utilisation des fractales en CGAO . . . . .	19
2.2	Systèmes Itérés de Fonctions (IFS) . . . . .	19
2.2.1	Création par Hutchinson . . . . .	19
2.2.2	Développement par Barnsley . . . . .	21
2.2.3	Calcul direct des points fixes . . . . .	21
2.2.4	Les produits tensoriels . . . . .	24
2.2.5	Calcul de la dimension fractale d'un IFS . . . . .	24
2.3	Les IFS barycentriques . . . . .	25
2.3.1	Génération des adresses dans l'espace des paramètres . . . . .	26
2.3.2	Génération d'une tessellation de l'attracteur dans l'espace barycentrique $\mathbf{BI}^N$ . . . . .	27
2.3.3	Calcul direct du point-fixe et de la tangente d'une transformation dans l'espace barycentrique . . . . .	27
2.3.4	Projection d'une combinaison barycentrique dans l'espace des points de contrôle . . . . .	29
2.3.5	Résumé de la méthode . . . . .	29
2.3.6	IFS barycentriques de courbes CGAO usuelles . . . . .	31
2.4	Automates d'IFS contrôlés (CIFS) . . . . .	36
2.4.1	Rappel sur les automates finis déterministes . . . . .	36
2.4.2	Construction des automates CIFS . . . . .	37
2.4.3	Opérations sur les automates . . . . .	38
<b>3</b>	<b>Schémas de subdivision</b>	<b>41</b>
3.1	Historique des schémas de subdivision . . . . .	41
3.1.1	Les schémas originels : Catmull-Clark et Doo-Sabin . . . . .	41
3.1.2	Quelques nouveaux schémas . . . . .	42
3.1.3	Pixar et la démocratisation des schémas de subdivision dans l'animation 3D . . . . .	42
3.1.4	Regain d'intérêt des surfaces subdivision . . . . .	42
3.1.5	Classification des schémas de subdivision . . . . .	43
3.2	Le schéma de subdivision de Catmull-Clark . . . . .	45
3.2.1	Formulation sous forme de masques . . . . .	45
3.2.2	Formulation sous forme de patches . . . . .	47
3.2.3	Formulation sous forme d'un automate CIFS . . . . .	51
3.3	Méthode de génération d'automate CIFS pour les schémas de subdivision . . . . .	53
3.3.1	Etape 1 : création des patches . . . . .	53

3.3.2	Etape 2 : génération des transformations, calcul des points fixes, et construction de l'automate	54
3.3.3	Etape 3 : génération des adresses . . . . .	54
3.4	Le schéma de Doo-Sabin . . . . .	55
3.4.1	Formulation sous forme de masques . . . . .	55
3.4.2	Formulation sous forme d'un automate CIFS . . . . .	56
3.5	Le schéma <i>Simplest</i> ou <i>Mid-edge</i> . . . . .	57
3.5.1	Formulation sous forme de masques . . . . .	57
3.5.2	Formulation sous-forme d'un automate CIFS . . . . .	58
3.6	Le schéma de Loop . . . . .	59
3.6.1	Formulation sous forme de masques . . . . .	59
3.6.2	Formulation sous forme d'un automate CIFS . . . . .	60
3.7	Le schéma <i>Honeycomb</i> . . . . .	61
3.7.1	Formulation sous forme de masques . . . . .	63
3.7.2	Formulation sous forme d'un automate CIFS . . . . .	64
3.8	Le schéma hybride de Catmull-Clark/Loop . . . . .	65
3.8.1	Formulation sous forme de masques . . . . .	66
3.8.2	Formulation sous forme d'un automate CIFS . . . . .	66
3.9	Les schémas de subdivision à support non-compact . . . . .	67
3.9.1	L'algorithme de génération des schémas de type B-Splines uniformes . . . . .	68
3.9.2	Les patches des schémas B-Splines uniformes . . . . .	68
<b>4</b>	<b>B-Splines Rationnelles Non-Uniformes (NURBS)</b>	<b>71</b>
4.1	Introduction aux NURBS . . . . .	72
4.1.1	Représentation et calcul des NURBS . . . . .	72
4.1.2	<i>Blossoming</i> (floraison) . . . . .	74
4.1.3	Insertion d'un noeud . . . . .	76
4.2	Stratégie numéro 1 : insertion au milieu de chaque intervalle nodal . . . . .	78
4.2.1	CIFS de dédoublement d'inter-noeuds . . . . .	79
4.2.2	Le vecteur inter-nodal . . . . .	79
4.2.3	Les polygones de contrôle . . . . .	79
4.2.4	Les automates CIFS de NUBS . . . . .	81
4.2.5	Automate CIFS de surfaces NUBS tensorielles de degré quelconque . . . . .	82
4.3	Stratégie numéro 2 : uniformisation du vecteur nodal . . . . .	84
4.3.1	Expression d'une étiquette uniforme en fonction du polygone de contrôle . . . . .	84
4.3.2	Uniformisation de surfaces NUBS tensorielles . . . . .	89
<b>5</b>	<b>Applications</b>	<b>91</b>
5.1	<i>Shaders</i> de combinaisons barycentriques pour le rendu temps réel . . . . .	91
5.1.1	Le <i>pipeline</i> des <i>shaders</i> en OpenGL . . . . .	93
5.1.2	Rendu de surfaces générées par un automate CIFS barycentrique . . . . .	95
5.1.3	Construction des différents automates CIFS pour le remplissage des <i>buffers</i> . . . . .	98
5.1.4	Comparaison avec les méthodes classiques de rendu temps réel . . . . .	100
5.1.5	Améliorations possibles . . . . .	103
5.2	MODITERE : un logiciel d'optimisation topologique . . . . .	104
5.2.1	Présentation du logiciel avant le début de la thèse . . . . .	104
5.2.2	Les apports de la thèse dans le modeleur MODITERE . . . . .	106
5.2.3	Vers un modèle géométrique générique à base de CIFS . . . . .	107
<b>6</b>	<b>Conclusion</b>	<b>111</b>
<b>A</b>	<b>Démonstration par Jean-Luc Baril</b>	<b>117</b>
<b>B</b>	<b>Notations</b>	<b>119</b>

# Table des figures

1.1	Exemple de courbe de Bézier cubique . . . . .	10
1.2	Comparaison entre une B-Spline uniforme et une B-Spline non-uniforme . . . . .	10
1.3	Différents exemples montrant les possibilités irréalistes des mondes virtuels . . . . .	11
1.4	Divers exemples de fractales. . . . .	11
1.5	Quelques exemples de fractales naturelles. . . . .	12
1.6	Exemple d'objets virtuels/naturels générés par fractales. . . . .	12
1.7	Plan du manuscrit représenté sous la forme de l'évolution des surfaces. . . . .	14
2.1	Visualisation de différents ensemble de Mandelbrot et de Julia. . . . .	18
2.2	Différents exemples d'utilisation de fractales en CGAO . . . . .	19
2.3	IFS du triangle de Sierpinskî . . . . .	20
2.4	Exemple de la génération d'une Fougère de Barnsley par le Jeu du Chaos . . . . .	21
2.5	Exemple du Triangle de Sierpinskî . . . . .	23
2.6	IFS correspondant au produit-tensoriel de deux ensembles de Cantor . . . . .	24
2.7	Représentation schématique de l'espace barycentrique de dimension 3, $\mathbf{BI}^3$ . . . . .	27
2.8	Projection d'une liste de combinaisons barycentriques dans différents triangles de contrôle. . . . .	29
2.9	Schéma représentant la méthode globale du calcul d'une tessellation de l'attracteur d'un IFS barycentrique pour un ensemble de contrôle donné. . . . .	30
2.10	Représentation sous forme de <i>blossoming</i> du calcul des coefficients d'une courbe de Bézier cubique . . . . .	31
2.11	Génération d'une courbe de Bézier cubique par l'algorithme de De Casteljau. . . . .	32
2.12	IFS barycentrique d'une courbe de Bézier quadratique. . . . .	33
2.13	Arbre de la construction par récurrence des fonctions de base B-Splines uniformes. . . . .	34
2.14	Exemple de l'application de l'algorithme de Chaikin . . . . .	34
2.15	Génération des B-Splines uniformes par l'algorithme de Lane-Riesenfeld . . . . .	35
2.16	IFS barycentrique d'une B-Spline quadratique. . . . .	36
2.17	Exemples de transformations contractantes modifiant la nature de l'objet. . . . .	37
2.18	Exemple d'un automate fini déterministe . . . . .	37
2.19	Exemple d'un automate CIFS incomplet composé de deux états et de huit transformations $\mathcal{T}_i$ . . . . .	38
2.20	Exemple d'automate CIFS complet. . . . .	38
2.21	Exemple de produit-tensoriel d'automates finis déterministes . . . . .	39
2.22	Exemple de raccord $C_0$ -continu au niveau de ses jonctions . . . . .	39
2.23	Structure globale d'un automate de raccord. . . . .	40
3.1	Exemple de l'application du schéma de Catmull-Clark sur le maillage "Tetris" . . . . .	42
3.2	Evolution du rendu des personnages humains dans les films d'animation Pixar. . . . .	43
3.3	Exemples de la subdivision topologique de faces régulières de plusieurs schémas. . . . .	43
3.4	Comparaison entre un schéma approximant et un schéma interpolant appliqués sur le même maillage de contrôle. . . . .	44
3.5	Exemples d'une itération sur un schéma primal et un schéma dual. . . . .	44
3.6	Exemples de surfaces limites obtenues à partir de deux schémas de continuités différentes . . . . .	45
3.7	Exemple d'une subdivision de Catmull-Clark sur un maillage quelconque. . . . .	46
3.8	Les masques de subdivision du schéma de Catmull-Clark. . . . .	47
3.9	Application de la méthode d'Halstead sur le schéma de Catmull-Clark . . . . .	48
3.10	Schéma illustrant la génération préalable de patches pour la méthode de Stam. . . . .	49
3.11	Résultat d'une succession d'itérations sur un patch irrégulier. . . . .	49
3.12	Décomposition d'un patch de Catmull-Clark régulier et d'un patch irrégulier en quatre sous-patches.	50

3.13 Application des transformations $\mathcal{T}_i$ dans l'espace des textures. . . . .	51
3.14 Algorithme de génération des adresses pour le schéma de Catmull-Clark. . . . .	52
3.15 Automate CIFS du schéma de Catmull-Clark . . . . .	53
3.16 Exemple d'une subdivision de Doo-Sabin sur un maillage de topologie quelconque. . . . .	55
3.17 Les masques du schéma de Doo-Sabin dans les cas régulier et irrégulier. . . . .	55
3.18 Deux exemples de patches de Doo-Sabin : régulier et irrégulier. . . . .	56
3.19 Automate CIFS du schéma de Doo-Sabin . . . . .	56
3.20 Les masques réguliers et irréguliers du schéma <i>Mid-edge</i> en une et deux itérations. . . . .	58
3.21 Création des sous-patches d'un patch irrégulier du schéma <i>Mid-edge</i> . . . . .	58
3.22 Automate CIFS du schéma <i>Mid-edge</i> . . . . .	58
3.23 Exemple d'une subdivision du schéma de Loop sur un maillage triangulaire quelconque. . . . .	59
3.24 Masques du schéma de subdivision de Loop. . . . .	60
3.25 Décomposition d'un patch régulier et d'un patch irrégulier de Loop en quatre sous-patches par application des règles de subdivision . . . . .	60
3.26 Automate CIFS du schéma de Loop . . . . .	61
3.27 Paramétrisation des morceaux de surface du schéma de Loop. . . . .	62
3.28 Algorithme de génération des adresses pour le schéma de Loop. . . . .	62
3.29 Exemple d'une subdivision du schéma <i>Honeycomb</i> sur un maillage quelconque. Les V-faces sont représentées en rouge et les F-faces en cyan. Remarquez que chaque F-face a le même nombre de sommets que la face dont elle est issue et que tous les nouveaux sommets sont de valence 3. . . . .	63
3.30 Les deux masques consécutifs du schéma <i>Honeycomb</i> . . . . .	63
3.31 Création des sous-patches à partir d'un patch régulier du schéma <i>Honeycomb</i> . . . . .	64
3.32 Les deux types de patches irréguliers du schéma <i>Honeycomb</i> . . . . .	64
3.33 Paramétrisation des morceaux de surface du schéma <i>Honeycomb</i> . . . . .	65
3.34 Automate CIFS du schéma <i>Honeycomb</i> . . . . .	65
3.35 Exemple d'une itération du schéma hybride Catmull-Clark/Loop . . . . .	65
3.36 Masques du schéma hybride Catmull-Clark/Loop . . . . .	66
3.37 Schéma des trois types de patches réguliers du schéma hybride de Catmull-Clark/Loop. . . . .	67
3.38 Les deux automates CIFS du schéma hybride de Catmull-Clark/Loop . . . . .	67
3.39 Génération de surfaces tensorielles B-Splines uniformes par la méthode du moyennage de faces. . . . .	68
3.40 Itération d'une surface de subdivision B-Spline par l'application de la méthode multiplication/moyennage. . . . .	69
3.41 Les patches de contrôle des schémas de subdivision de type B-Splines générés par l'algorithme de Zorin. . . . .	70
4.1 Arbre de la construction par récurrence des fonctions de base des B-Splines cubiques non-uniformes. . . . .	73
4.2 Comparaison entre la non-uniformité et la rationalité. . . . .	74
4.3 Exemple de <i>blossoming</i> sur une B-Spline quadratique non-uniforme . . . . .	75
4.4 <i>Blossoming</i> de l'insertion d'un noeud entre les bornes d'une NUBS quadratique . . . . .	76
4.5 <i>Blossoming</i> de l'insertion d'un nouveau noeud dans un intervalle nodal où la courbe n'est pas définie . . . . .	77
4.6 Différents exemples des conséquences d'une insertion de noeuds dans un vecteur nodal . . . . .	77
4.7 Schéma des transformations $\mathcal{L}$ et $\mathcal{R}$ sur un vecteur inter-nodal quadratique. . . . .	79
4.8 <i>Blossoming</i> permettant l'expression d'un nouveau polygone de contrôle quadratique en fonction de l'ancien. . . . .	79
4.9 Calcul des nouveaux polygones de contrôle en fonction des anciens pour les NUBS quadratiques. . . . .	80
4.10 <i>Blossoming</i> des nouveaux points de contrôle en fonction des anciens pour les NUBS cubiques. . . . .	80
4.11 Matrices de l'automate CIFS de NUBS cubique. . . . .	80
4.12 Construction de l'automate CIFS de NUBS quadratiques . . . . .	81
4.13 Automate CIFS de NURBS cubique. . . . .	82
4.14 Construction de l'automate CIFS de surfaces NUBS tensorielles bi-quadratiques . . . . .	83
4.15 Automate CIFS de surfaces NUBS tensorielles bi-cubiques . . . . .	83
4.16 <i>Blossoming</i> de l'uniformisation d'une NUBS quadratique . . . . .	84
4.17 Uniformisation d'une NUBS quadratique par <i>blossoming</i> . . . . .	85
4.18 <i>Blossoming</i> exprimant un point de la NUBS en fonction des points de contrôle en passant par un polygone de contrôle intermédiaire uniforme . . . . .	85
4.19 Expression d'une étiquette cubique quelconque en fonction du polygone de contrôle par <i>blossoming</i> . . . . .	88
4.20 Uniformisation d'une NUBS quadratique à sept morceaux . . . . .	89
4.21 Expression d'un étiquette-double biquadratique quelconque par <i>blossoming</i> . . . . .	90

5.1	Schéma du <i>pipeline</i> OpenGL dans le moteur de rendu par <i>shaders</i> de combinaisons barycentriques . . . . .	92
5.2	Un exemple d' <i>abstract patch</i> quadrangulaire et un exemple triangulaire. . . . .	94
5.3	Comparaison entre une tessellation par face et par arête . . . . .	96
5.4	Exemple des différents cas de <i>trimming</i> par le <i>Geometry Shader</i> . . . . .	97
5.5	Quelques exemples de <i>trimming</i> obtenu par approximation barycentrique à différents niveaux de subdivision . . . . .	97
5.6	Comparaison entre les deux méthodes de calcul des normales pour différents niveaux de tessellation. . . . .	98
5.7	Différents exemples de topologie irrégulières de patches. . . . .	99
5.8	Quelques exemples de paramétrisation de morceaux de surface limite. . . . .	99
5.9	Comparaison entre les trois méthodes de modélisation géométrique des surfaces de subdivision en fonction du temps de calcul et de la mémoire occupée sur un maillage animé. . . . .	104
5.10	Différents exemples de rendu par l'utilisation de <i>shaders</i> de combinaisons barycentriques . . . . .	105
5.11	Deux exemples de réalisations faites sur le logiciel MODITERE. . . . .	106
5.12	Différentes coupes de l'échangeur thermique conçu sur le logiciel MODITERE . . . . .	107
5.13	Quelques exemples de réalisations impliquant des NURBS sur le logiciel MODITERE du LIB . . . . .	108

## Avant propos

L'ensemble des images présentes dans ce manuscrit appartiennent à leurs auteurs respectifs. Par respect de la propriété intellectuelle, il a été indiqué leur provenance mais il se peut que certaines sources aient été oubliées, ne m'en voulez pas! Pour les images de ma propre création, elles sont toutes sous licence **Creative Commons**, vous êtes donc libres de réutiliser et de modifier ces images comme bon vous semble.

Lors de l'écriture de ce manuscrit, des couleurs ont été mises dans la plupart des images et même parfois dans le texte. Il est donc fortement conseillé de lire ce manuscrit en couleur. Vous êtes prévenus! Il est également conseillé d'aller détacher la dernière page ([Annexe B](#)) qui contient un résumé des notations utilisées dans la thèse et de la garder sous la main lors de la lecture du manuscrit.

Bonne lecture!

# Chapitre 1

## Introduction

### 1.1 Une histoire de courbes et de surfaces

#### 1.1.1 Les courbes de Bézier et l'algorithme de Casteljau

La conception de ces courbes a été faite parallèlement par deux mathématiciens français, ingénieurs concurrents dans l'industrie de l'automobile : Paul de Faget de Casteljau (Citroën) en 1959 et Pierre Bézier (Renault) en 1962. Pour des raisons de secret industriel, ces résultats ne font pas l'objet d'une publication au moment de leur découverte et c'est après coup que les **courbes** sont nommées **de Bézier**, et l'**algorithme** pour les construire **de Casteljau**.

Dans les années 50-60, tout ce qui est pièce mécanique d'une voiture suit déjà des contraintes très précises avec des tolérances exprimées en centièmes de millimètre. Par exemple, la construction d'un moteur de voiture ne peut se permettre des approximations trop conséquentes. Au contraire, pour ce qui concerne la carrosserie, les contraintes sont quasiment inexistantes : le seul juge est le *designer* dont l'oeil artistique, mais surtout subjectif, sur l'esthétique de la carrosserie varie avec le temps et l'humeur. Depuis les croquis du *designer* jusqu'à la vérification finale par les contrôleurs avant la mise en circulation du véhicule, chaque service par lequel passent les plans s'autorise de petites modifications qui permettent parfois d'éviter de consommer un peu trop d'étain sur une soudure (et donc de limiter les coûts de matière première) et parfois de plus facilement encastrer une portière. Après tout, chacun peut déformer un peu la carrosserie, ce n'est pas les quelques millimètres de différence qui vont changer de manière conséquente l'esthétique ou l'aérodynamisme.

Pour éviter toutes ces petites adaptations qui brisent la cohérence du modèle, Bézier et Casteljau ont pour objectif la conception d'un outil permettant à la fois de dessiner des surfaces de manière intuitive pour un *designer* non-mathématicien et de construire par la suite ces mêmes surfaces pour les ingénieurs et techniciens. Pour le citer, Bézier cherche "*une définition unique et indiscutable [de la carrosserie], établie par le styliste lui-même et transmise ensuite sous forme numérique à tous les groupes intervenants dans les processus jusqu'au contrôleur opérant à la sortie de la chaîne de fabrication*". La solution trouvée par les deux ingénieurs est une courbe définie par un ensemble ordonné de points de contrôle appelé **polygone de contrôle**. Les deux points extrêmes de ce polygone sont interpolés (c'est à dire que la courbe passe par ces points) et les points intermédiaires permettent de donner une allure générale à la courbe (cf [Figure 1.1](#)). Plus le nombre de points intermédiaires est important, plus le degré de la courbe est élevé. La notion de courbe est facilement étendue à la notion de surface en définissant celle-ci comme le produit-tensoriel de deux courbes. Dans ce cas, l'ensemble des points de contrôle n'est plus un polygone mais une **grille de contrôle**.

Par la suite, les courbes de Bézier ont été remplacées en **Conception Géométrique Assistée par Ordinateur (CGAO)** par les **B-Splines Rationnelles Non-Uniforme (NURBS)** mais les courbes de Bézier cubiques sont toujours utilisées dans d'autres types d'applications. Par exemple, pour tracer des courbes interpolantes dans les logiciels de dessin par ordinateur (*e.g.* Inkscape, utilisé pour la création de la plupart des schémas de ce manuscrit) et pour le contrôle de l'évolution d'une animations entre deux *key-frames* dans les logiciels d'Infographie 3D et d'Animation (*e.g.* Blender, utilisé pour la conception des modèles 3D en exemples).

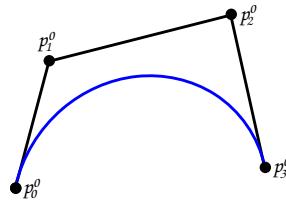


FIGURE 1.1 – Exemple d'une courbe de Bézier cubique (en bleu) définie par un polygone de contrôle (en noir). Les points extrêmes  $p_0^0$  et  $p_3^0$  sont interpolés et les points intermédiaires  $p_1^0$  et  $p_2^0$  donnent à la courbe son allure.

### 1.1.2 L'ingénierie : les B-Splines et les NURBS

Les courbes de Bézier sont rapidement remplacées par les **B-Splines non-uniformes** [CS66] (elles même issues des **B-Splines uniformes** [Sch46]) qui permettent la construction d'une courbe dont le nombre de points de contrôle peut augmenter tout en conservant un degré fixe (voir exemples en Figure 1.2). En effet, une B-Spline définie par un **polygone de contrôle** est composée de plusieurs morceaux définis chacun par un sous-polygone de contrôle dont le nombre de points dépend du degré de la courbe. La continuité de la courbe au niveau des jonctions entre les morceaux est assurée par des points de contrôle communs à plusieurs sous-polygones. A ce polygone de contrôle est ajouté un **vecteur nodal** qui agit comme un paramètre de tension sur la courbe. Tout comme les surfaces de Bézier, les surfaces B-Splines sont définies par une grille de contrôle.

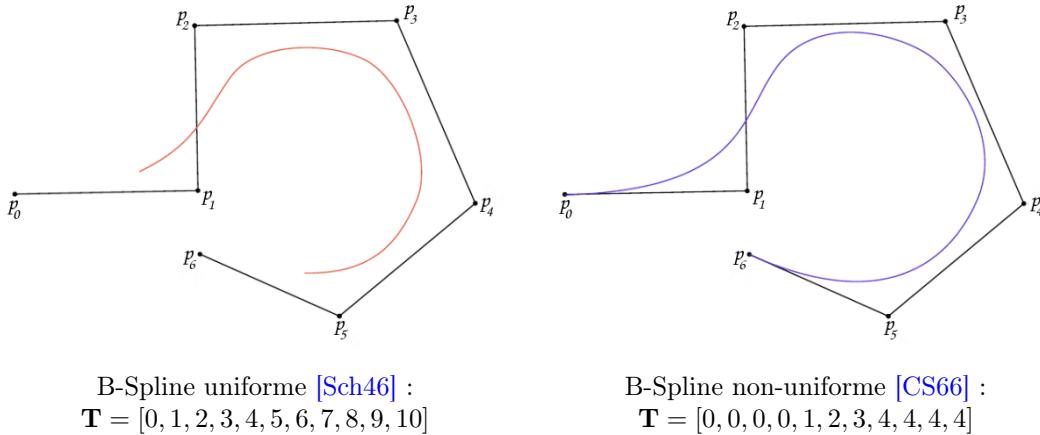


FIGURE 1.2 – Comparaison entre une B-Spline uniforme et une B-Spline non-uniforme. Dans le cas d'une B-Spline uniforme, le vecteur de noeud  $\mathbf{T}$  est composé de noeuds également espacés ce qui entraîne qu'il est rarement précisément bien que présent. Dans le cas d'une B-Spline non-uniforme, ce vecteur nodal influe la forme de la courbe et est, par conséquent, toujours précisément.

En 1972, Maurice G. Cox généralise ces courbes aux **B-Splines Rationnelles Non-Uniformes (NURBS)** [Cox72]. Celles-ci permettent un plus grand contrôle de la forme que les B-Splines non-uniformes et notamment la possibilité de construire des coniques. C'est grâce à cela qu'elles se sont imposées comme le standard dans les systèmes de CGAO et le restent encore aujourd'hui. Un grand nombre d'outils, devenus par la suite indispensables en CGAO, sont conçus spécialement pour les NURBS ce qui impose à tout nouveau formalisme voulant se faire une place dans le domaine de proposer également cette palette d'outils.

### 1.1.3 Le monde virtuel : les surfaces de subdivision

En 1974, George Merrill Chaikin montre que les B-Splines uniformes de degré 2 peuvent être construites par un algorithme itératif de la même manière que l'algorithme de Casteljau construit les courbes de Bézier [Cha74]. En 1978, les deux binômes que sont Edwin Earl Catmull et James Henry Clark [CC78] et Daniel Doo et Malcolm Sabin [DS78] généralisent la méthode itérative de Chaikin pour définir une extension des B-Splines uniformes permettant d'avoir comme ensemble de contrôle un maillage qui n'est pas forcément une grille : les **surfaces de subdivision**.

En s'extrayant de la contrainte imposée par le produit-tensoriel, les surfaces de subdivision sont plus intuitives à manipuler et permettent une grande flexibilité. Malgré ces avantages, elles ne sont que très rarement utilisées dans les systèmes de CGAO industriels car elles n'offrent pas les mêmes possibilités de contrôle que les NURBS.

Il faut attendre 1997 pour qu'elles deviennent une référence, mais dans le domaine des films d'animation, grâce au film de Pixar : "Le Jouer d'échecs" (cf. Figure 1.3 gauche) et les travaux de Tony DeRose, Michael Kass, et Tien Truong [DKT98]. En effet, dans le cadre d'un univers entièrement virtuel, les formes n'ont pas besoin d'être constructibles, elles peuvent même être physiquement irréalisables comme dans les œuvres de l'artiste néerlandais Maurits Cornelis Escher, car dans le virtuel, aucune loi physique ne les constraints (cf. Figure 1.3 centre). C'est d'autant plus valable dans l'univers du *cartoon* où les corps se déforment exagérément pour donner plus de dynamisme (cf. Figure 1.3 droite), car ce qui compte est l'action et non son réalisme, comme énoncé par les animateurs de Disney : Ollie Johnston et Frank Thomas dans les 12 principes de base de l'animation du livre : *The Illusion of Life*. Les graphistes 3D et les animateurs ont besoin d'un modèle sans contrainte pour laisser libre cours à leur imagination et les possibilités offertes par les surfaces de subdivision correspondent parfaitement à ce besoin de liberté.

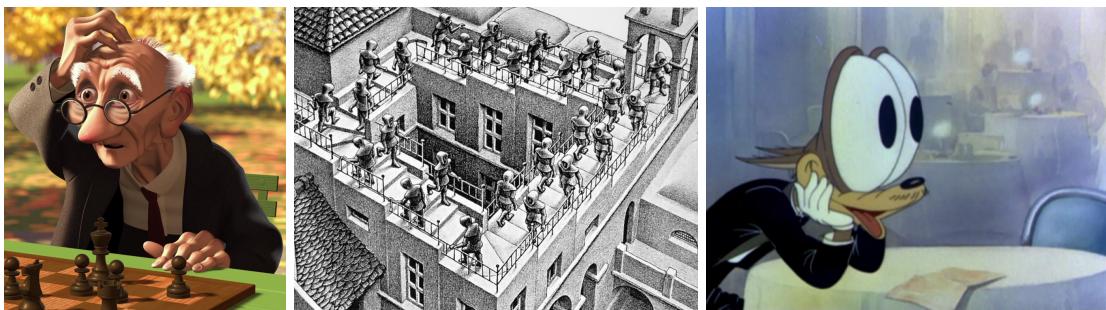


FIGURE 1.3 – A gauche, Geri, un des premiers personnages représentés par des surfaces de subdivision. Au centre, l'escalier du monastère de la gravure d'Escher : "Montée et Descente" qui est physiquement irréalisable. A droite, le loup de Tex Avery dont les yeux s'étirent outre mesure pour mieux voir le Chaperon dans *Red Hot Riding Hood*.

#### 1.1.4 Les fractales

A la fin du XIX<sup>e</sup> et au début du XX<sup>e</sup> siècle, plusieurs mathématiciens européens décrivent des figures construites de manière itérative dont les propriétés sont paradoxales pour les mathématiques de l'époque, ce qui leur vaut le surnom de "monstres mathématiques", mais elles sont maintenant connues sous le nom de **fractales**. Par exemple, l'italien Giuseppe Peano [Pea90] et l'allemand David Hilbert [Hil91] définissent chacun une courbe différente qui permet de remplir un carré, ainsi la courbe censée être de dimension 1 comme n'importe quelle autre courbe est de dimension 2. Le mathématicien allemand Georg Cantor construit de manière itérative un sous-ensemble de la droite des réels de mesure nulle mais qui n'est pas pour autant dénombrable [C<sup>t</sup>84]. A l'inverse, le suédois Helge von Koch, définit une courbe de longueur infinie qui pourtant délimite une surface bornée, ainsi qu'un flocon dont le périmètre est infini mais dont l'aire converge vers un nombre fini [Koc04]. Le polonais Waclaw Sierpiński et l'autrichien Karl Menger étendent l'espace de Cantor dans le triangle et le tapis de Sierpiński (1915 et 1916) et dans l'éponge de Menger [Men26]. Plusieurs exemples de fractales citées dans ce paragraphe sont présents dans la Figure 1.4.

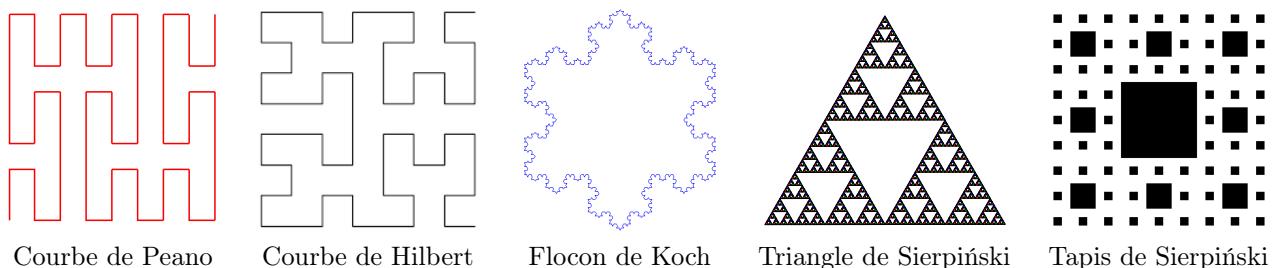


FIGURE 1.4 – Divers exemples de fractales.

Dans les années 60, le franco-américain Benoît Mandelbrot, alors chercheur sur la transmission optimale dans les milieux bruités pour IBM, s'intéresse à ces formes étranges qui tiennent en échec les mathématiciens. À partir des travaux de l'allemand Felix Hausdorff [Hau18] sur les dimensions non-entières et de la conviction que ces objets doivent être abordés du point de vue de l'homothétie d'échelle, il construit un modèle mathématique qui permet à la fois de construire ces objets mais également d'en étudier les propriétés. Il invente alors le terme fractal issu du latin *fractus* (brisé, irrégulier), nomme ces formes particulières les **fractales**, et définit la notion d'**auto-similarité** : "chaque partie est un tout, mais en plus petit". Dans son livre, *Les Objets fractals - Forme, hasard et dimension* [Man75], il montre non seulement que son formalisme s'adapte aux objets précédemment définis par ses collègues mathématiciens, mais également que des objets, dits non-géométriques jusqu'alors, était également des fractales. Par exemple, elles existent à l'état naturel : chaque florette du chou romanesco est une représentation à une échelle plus petite du chou lui-même, chaque morceau de nuage est un nuage plus petit (voir Figure 1.5 gauche et centre). À une échelle microscopique, le mouvement brownien qui décrit le déplacement, supposé aléatoire, d'une particule qui entre en collision avec d'autres suit en réalité un déplacement fractal (voir Figure 1.5 droite).



FIGURE 1.5 – Quelques exemples de fractales naturelles.

En 1981, John Hutchinson [Hut81] exploite l'auto-similarité des fractales pour concevoir les **Systèmes Itérés de Fonctions (IFS)**. Ces systèmes, composés d'un ensemble de transformations contractantes, sont appliqués sur un compact de manière itérative. Après une infinité d'itérations, le système converge vers un attracteur qui est une fractale, et ce quel que soit le compact de départ. Ainsi, la fractale est définie uniquement par ses propriétés d'auto-similarité représentées par les transformations contractantes. Par la suite, Michael Fielding Barnsley [Bar14] développe le formalisme, en particulier en démontrant le **Théorème du collage** qui énonce que tout ensemble compact de points peut être approché par l'attracteur d'un IFS. Il crée également le **Jeu du Chaos** (cf. Figure 2.4 page 21) qui permet d'afficher algorithmiquement, et de manière optimisée, l'attracteur sur un écran d'ordinateur.

Les fractales sont utilisées dans la conception d'univers virtuels et en particulier lors de génération procédurale. En effet, le Jeu du Chaos de Barnsley permet de générer de manière déterministe et à un coût faible en temps de calcul et en mémoire des formes fractales sur ordinateur. Par exemple, il peut générer des effets de particules tels que les flammes, les nuages ou la végétation. Il est également utilisé pour "colorer" des *displacement maps* (textures qui modifient la géométrie en enfonçant ou surélevant des points) qui seront ensuite plaquées sur des surfaces planes pour faire apparaître des montagnes ou des vagues sur l'océan. Quelques exemples d'objets purement virtuels générés par des techniques basées sur des fractales sont données en Figure 1.6.



FIGURE 1.6 – Exemple d'objets virtuels/naturels générés par fractales.

Du côté de la CGAO, ce sont les propriétés topologiques intrinsèques (telles que l'auto-similarité) des fractales qui ont permis de résoudre de manière élégante certains problèmes complexes comme celui de l'optimisation topologique. L'idée d'utiliser des fractales plutôt que des objets plus classiques est d'avoir des surfaces ou des volumes non-pleins sans pour autant sacrifier la résistance de l'objet. L'intérêt de volumes à trous est par exemple d'économiser de la matière première, d'alléger la structure, et/ou d'évacuer de manière plus efficace la chaleur. La présence de fractales dans l'architecture est bien antérieure aux travaux de Mandelbrot, par exemple les propriétés de celles-ci avaient déjà sollicitées l'intérêt de l'ingénieur dijonnais, Gustave Eiffel, lors de la construction d'une célèbre tour parisienne.

## 1.2 Objectif

Comme décrit plus haut, chaque corps de métier de l'industrie de la CGAO a un objectif différent et donc utilise des outils spécifiques, adaptés aux différentes contraintes, pour atteindre ce but. Par exemple, le *designer* utilise des **surfaces de subdivision**, très instinctives d'utilisation pour laisser libre cours à l'inspiration sans se préoccuper du modèle mathématique sous-jacent. Quant à l'ingénieur chargé de construire ce modèle, il privilégie l'utilisation des **NURBS** dont le contrôle est certes beaucoup moins intuitif, car il demande des connaissances mathématiques précises, mais est aussi beaucoup plus précis et permet de réaliser une batterie de simulations numériques sur le modèle : constructibilité, résistance, aérodynamisme... Enfin, un autre ingénieur chargé de l'optimisation des volumes, que ce soit pour économiser de la matière ou résoudre des contraintes thermo-dynamiques, utilise plutôt de son côté des **structures lacunaires fractales** dont la complexité permet de trouver des solutions nouvelles. Toutes ces personnes construisent le même modèle et pourtant elles n'utilisent pas les mêmes outils ; elles travaillent ensemble mais ne parlent pas la même langue.

La solution la plus courante actuellement pour pallier cette "barrière de la langue" est la conversion. Le modèle est converti chaque fois qu'il change de service mais comme il n'y a pas d'équivalences strictes entre les différents modèles, chaque conversion entraîne une détérioration et une complexification du modèle à cause des différentes approximations nécessaires. Et ces conversions peuvent être nombreuses car il arrive souvent que le modèle fasse des aller-retours entre les services. De plus, les conversions sont coûteuses, et même si des systèmes de conversion automatique ont été définis [SKSD14, SKSD16], celles-ci sont toujours supervisées par un ingénieur spécialisé.

Une autre solution, qui est celle que nous allons exploiter dans ce manuscrit et également celle que Bézier préconisait déjà il y a 60 ans, est l'utilisation d'une représentation commune pour tout le monde : il faut que le *designer* qui imagine la voiture et l'ingénieur en charge des simulations d'aérodynamisme parlent la même langue, mais il ne faut pas pour autant imposer à l'un la langue de l'autre. En effet, il n'est pas question de perdre l'expertise métier du *designer* en changeant son outil de travail intuitif que sont les surfaces de subdivision par un outil beaucoup plus difficile à appréhender, car mathématiquement complexe, que sont les NURBS. Il n'est pas non plus raisonnable d'espérer que les surfaces de subdivision puissent offrir, au moins à l'heure actuelle, les mêmes possibilités que les NURBS.

Des solutions ont déjà été proposées en ce sens : Dodgson et Kosinka proposent une interface graphique intuitive permettant au *designer* de travailler avec des NURBS "sans le savoir" [DK16] mais le modèle visuellement simple du point de vue de l'utilisateur peut s'avérer très complexe dans sa représentation géométrique réelle sous-jacente. Ma [Ma05] et Antonelli *et al.* [ABC<sup>+</sup>13] proposent la conception d'un moteur de CGAO basé sur les surfaces de subdivision mais celui-ci reste théorique car de nombreux outils qui constituent le noyau des moteurs CGAO (*geometric kernel*) n'existent actuellement que pour les NURBS. Se pose également le problème de la rétro-compatibilité : comment exploiter dans ce nouveau moteur tous les modèles déjà conçus ?

Un autre type de solution est la création de schémas de subdivision non-uniformes (NURSS) [SZSS98, MRF06] qui auraient à la fois les avantages des surfaces de subdivision (topologie arbitraire) et des NURBS (contrôle précis de la forme) mais ceux-ci sont restreints aux bas degrés (2 ou 3). Cashman *et al.* [CADS09], ont conçu des NURBS de degré impair à la topologie arbitraire (qui peuvent s'interpréter comme des NURSS de haut-degré) mais leur modèle se heurte au problème de l'inertie industrielle : pourquoi changer notre méthode qui fonctionne pour une autre même si elle est meilleure ?

Notre approche est différente, nous cherchons à concevoir un **formalisme global** qui gère à la fois les **surfaces de subdivision**, les **NURBS**, et les **fractales** qui sont généralement exclues de la discussion. Notre position est la suivante : les fractales sont des formes auto-similaires, tout comme les B-Splines uniformes qui sont à l'origine des surfaces de subdivision et des NURBS. Est-ce que ces deux types de surfaces ne seraient pas elles aussi auto-similaires et donc représentables de manière itérative ? Ainsi, il existerait un formalisme général, basé sur une représentation itérative de tous les objets, qui gérerait à la fois les surfaces de subdivision, les NURBS, et les fractales. Ce formalisme que nous proposons est l'utilisation d'automate de Systèmes Itérés et Contrôlés de Fonctions (CIFS).

En résumé : *"Notre objectif est de concevoir un modèle géométrique commun utilisable par tous les corps de métiers tout en conservant le savoir-faire de chacun. Ainsi, notre modèle doit être capable de gérer à la fois les surfaces de subdivision dont la simplicité d'utilisation permet la liberté créative au designer, les NURBS dont la précision des calculs et le grand contrôle permet à l'ingénieur de résoudre les contraintes sans détruire l'aspect esthétique, et les fractales dont la complexité de structure permet une optimisation des volumes pour une économie de matière première ou de nouvelles solutions à certaines contraintes."*

### 1.3 Organisation générale du manuscrit

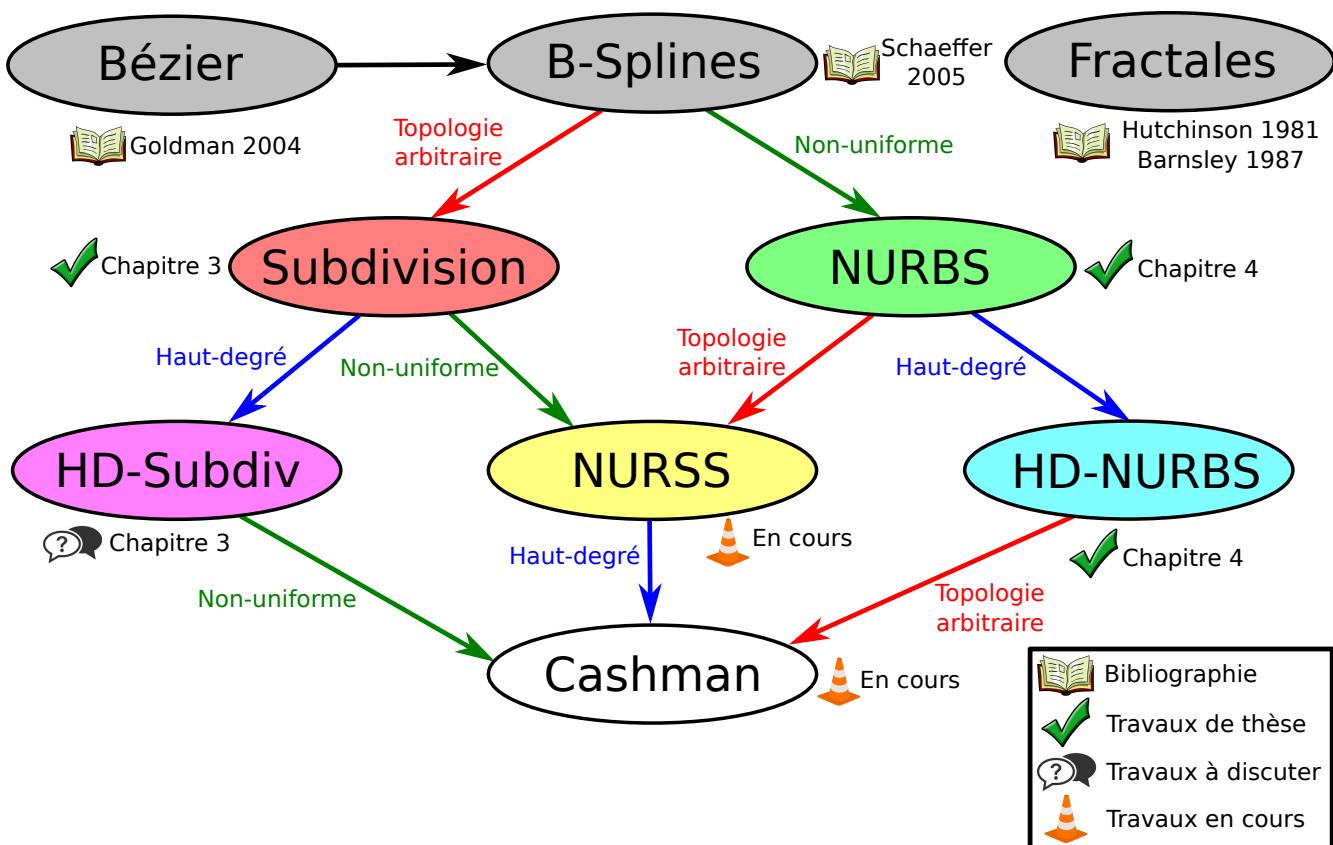


FIGURE 1.7 – Plan du manuscrit représenté sous la forme de l'évolution des surfaces. Les travaux antérieurs sont indiqués par un livre et expliqués dans le [Chapitre 2](#). Les travaux originaux terminés et en cours de réflexion sont marqués respectivement d'un *check* et d'une bulle de discussion et le numéro du chapitre correspondant est indiqué.

Le formalisme des automates CIFS qui est utilisé tout au long de ce manuscrit est exposé dans le [Chapitre 2](#). Pour commencer, les fractales et les outils classiques utilisés pour les étudier sont décrits. Ensuite, les Systèmes Itérés de Fonctions (IFS) sont présentés ainsi que les contraintes imposées sur les fonctions de ces systèmes pour assurer la convergence. Des travaux antérieurs exploitant les IFS barycentriques pour générer des courbes et surfaces de CGAO sont utilisés à titre d'exemple. Pour finir, le formalisme des CIFS et les différents outils dont il permet l'utilisation sont présentés.

Dans le [Chapitre 3](#) sont présentés les automates CIFS des schémas de subdivision les plus courants. Pour commencer, un rapide historique des schémas de subdivision est dépeint, puis les notions communes à tous les schémas sont exposées ainsi qu'une méthode de classification. Ensuite, chaque schéma est présenté de deux manières différentes : pour commencer sous la forme classique de règles de subdivision et ensuite sous forme d'un automate CIFS qui est la représentation originale de ces différents schémas développée au cours de cette thèse. La représentation sous forme d'automates CIFS des schémas de Catmull-Clark, Doo-Sabin, Loop et *Mid-edge* a déjà été publiée [[MNLG18b](#)] ; les autres automates énoncés dans ce chapitre sont inédits.

Le [Chapitre 4](#) contient deux méthodes permettant de représenter une surface NURBS tensorielle de degré quelconque en tant qu'automate CIFS. La première méthode utilise la représentation par *blossoming* des NURBS de degré 2 et 3 pour trouver la stationnarité du processus de subdivision et en déduire les automates CIFS correspondants. A partir de ces deux exemples, cette méthode est généralisée pour construire l'automate correspondant à une NURBS de degré quelconque. Ensuite, la technique du produit-tensoriel d'automates est utilisée pour générer les automates de surfaces NURBS tensorielles à partir des automates de courbes. Ces résultats font l'objet de l'article [[MGL<sup>+</sup>19](#)]. La seconde méthode utilise également le *blossoming* mais cette fois pour démontrer que chaque morceau d'une NURBS, pris indépendamment des autres, est uniforme pour un polygone de contrôle précis. Les formules permettant de calculer ce polygone de contrôle uniforme sont directement issues de la formulation *blossoming*. Une fois la NURBS uniformisée, l'automate CIFS permettant de la construire est simplement celui des B-Splines uniformes. La méthode est également étendue aux surfaces générées par produit tensoriel en utilisant un *blossoming* de surface. Cette méthode fait l'objet d'un article en cours de rédaction [[MGLN19](#)].

Les nouvelles possibilités apportées par ce nouveau formalisme sont présentées dans le [Chapitre 5](#). En particulier, deux applications sont décrites. La première est une implémentation du formalisme CIFS (en particulier pour les surfaces de subdivision) exploitant la puissance des cartes graphiques modernes. Celle-ci permet un affichage temps réel et à la volée de la surface (*i.e.* l'attracteur), directement à partir du maillage de contrôle, sans les habituelles étapes intermédiaires de subdivision. De ce fait, une économie de mémoire est faite par rapport au rendu de maillages pré-subdivisés et un gain de temps est réalisé par rapport aux subdivisions classiques. La seconde est l'utilisation du logiciel pré-existant MODITERE pour la gestion des interactions entre des objets de différentes natures (surfaces de subdivision, NURBS, et fractales). La nouvelle représentation des surfaces de subdivision et des fractales sous la forme d'automates CIFS permet ces interactions, jusqu'alors trop difficiles à mettre en oeuvre, voire impossible, comme la simple utilisation d'outils déjà présents dans le formalisme des CIFS et déjà implémentés dans le logiciel.



## Chapitre 2

# Systèmes Itérés (et Contrôlés) de Fonctions

Dans ce chapitre sont présentés les automates de Systèmes Itérés et Contrôlés de Fonctions (CIFS). Dans un premier temps, les fractales sont présentées de la manière habituelle puis sous la forme moins courante de Systèmes Itérés de Fonction (IFS). Ensuite, il est montré qu'en exploitant les IFS dans l'espace barycentrique, il est possible de construire certaines courbes et surfaces de CGAO classiques comme les surfaces de Bézier et les surfaces B-Splines. Par la suite, les automates CIFS qui sont une extension du formalisme des IFS à des automates, et qui sont également le formalisme au coeur de cette thèse, sont présentés. Pour finir, certains outils utilisés pour les CIFS, tel que le raccord entre deux objets représentés sous la forme d'automate CIFS, sont décrits.

## 2.1 Fractales

### 2.1.1 Ensembles de Julia, de Fatou, et de Mandelbrot

Dans les années 60, Benoit Mandelbrot étudie des objets mathématiques définis très simplement mais dont les propriétés sont très complexes. Il travaille en particulier sur les ensembles définis par les mathématiciens français Gaston Julia et Pierre Fatou : les ensembles complémentaires de Julia et de Fatou ainsi qu'un autre ensemble qui prendra le nom de Mandelbrot.

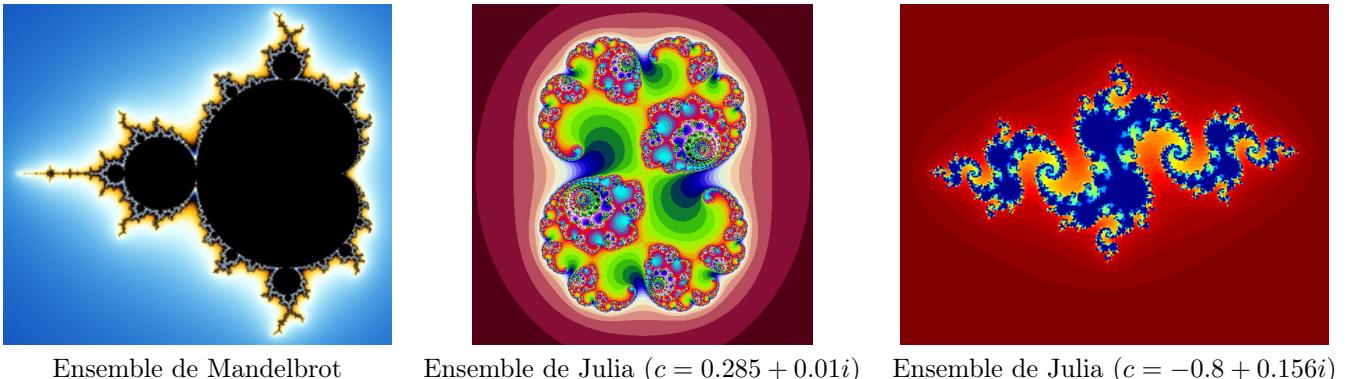
L'**ensemble de Mandelbrot** est une fractale définie comme l'ensemble des points  $c$  du plan complexe pour lesquels la suite suivante est bornée :

$$\begin{cases} z_0 = 0 \\ z_{n+1} = z_n^2 + c \end{cases}$$

Pour tout  $z_n$  dont le module est strictement supérieur à 2, la suite diverge. Cette propriété est utilisée pour afficher informatiquement l'ensemble : chaque pixel représente un  $c$  différent et la suite est itérée. Dès que le complexe obtenu  $z_n$  est de module strictement plus grand que 2, le pixel est colorié (en général la couleur dépend du nombre d'itérations). Si un nombre maximal d'itérations est atteint la suite est considérée comme bornée et le pixel correspondant reste noir. Un exemple est présent en [Figure 2.1](#) (gauche).

A l'inverse de l'ensemble de Mandelbrot qui fixe  $z_0$  et étudie les variations de  $c$ , l'**ensemble de Julia** fixe  $c$  et étudie les différents points de départ possible. Pour visualiser l'ensemble de Julia, chaque pixel est considéré comme un point de départ et la suite est itérée. Si le module dépasse 2, le pixel et tout ceux par lesquels la suite est passée sont coloriés (la couleur dépendant du nombre d'itérations avant la divergence). Deux exemples sont donnés en [Figure 2.1](#).

Pour chaque point  $c$  appartenant à l'ensemble de Mandelbrot, l'ensemble de Julia correspondant à ce  $c$  est connexe. Ainsi, l'ensemble de Mandelbrot peut également être défini comme l'ensemble des complexes pour lesquels l'ensemble de Julia correspondant est connexe.



Ensemble de Mandelbrot

Ensemble de Julia ( $c = 0.285 + 0.01i$ )Ensemble de Julia ( $c = -0.8 + 0.156i$ )

FIGURE 2.1 – Visualisation de différents ensemble de Mandelbrot et de Julia.

En représentant visuellement ces objets, il apparaît que certaines parties de ceux-ci se répètent. Plus précisément, ils sont composés d'un ensemble de sous-parties identiques à l'objet lui-même. Mandelbrot aborde ces objets, qu'il nomme **fractales**, du point de vue de cette répétition appelée **l'auto-similarité**. Une fractale  $\mathcal{F}$  est définie comme l'union de fractales identiques à la première près  $\mathcal{F}_i$  :

$$\mathcal{F} = \bigcup_{i=0}^{n-1} \mathcal{F}_i$$

### 2.1.2 Dimensions fractales

Contrairement à la plupart des objets géométriques, les fractales sont des objets de dimension non-entière. Dans [Man67], qui est souvent considéré comme le point de départ de la création du modèle des fractales, Mandelbrot pose la question suivante : quelle est la longueur de la côte de Grande-Bretagne ? Il est facile d'en mesurer une approximation mais une mesure précise se heurte au **paradoxe de la ligne de côte**. Celui-ci énonce que plus l'outil de mesure est précis, plus la longueur mesurée est grande. Imaginez un piéton et un crabe qui marchent sur la ligne de côte de manière à ce que chaque pas soit strictement sur la ligne brisée qui sépare la terre de l'eau, avec le niveau de la mer supposé fixe. La distance parcourue par le crabe, dont les pas sont plus petits que les enjambées du piéton, sera plus grande que celle du marcheur. L'idée est que plus l'outil de mesure est précis, plus il va être influencé par la moindre aspérité de cette ligne brisée et donc plus la longueur va être grande ; celle-ci est donc infinie du point de vue des méthodes de mesure traditionnelles.

La solution proposée par Mandelbrot, inspirée par les travaux de Hausdorff [Hau18] est la suivante :

- doubler la taille d'un segment (de dimension 1) équivaut à multiplier par  $2^1$  sa longueur
- doubler la taille d'un carré (de dimension 2) équivaut à multiplier par  $2^2$  son aire
- doubler la taille d'un cube (de dimension 3) équivaut à multiplier par  $2^3$  son volume.

Soit un objet dont la mesure est multipliée par  $k$  lorsque sa taille est multipliée par  $n$ , sa dimension est  $\mathbf{d} = \log_n(k)$ . Par exemple, la courbe de Koch quadruple sa mesure chaque fois que sa taille triple, elle est donc de dimension  $\log_3(4)$ .

Il n'est pas toujours possible de simplement réduire une fractale à un facteur de taille et un facteur de mesure, ce qui fait que le calcul de la dimension d'une fractale n'est pas toujours aussi aisés. Les dimensions les plus courantes sont :

- la dimension de Hausdorff [Hau18], définie quel que soit l'ensemble mais difficile à calculer
- la dimension de Minkowski-Bouligand (ou *box-counting*) qui consiste simplement à compter le nombre d'éléments de taille fixes nécessaires pour recouvrir un ensemble.

Lorsque les fractales sont représentées sous la forme de Systèmes de Fonctions Itérés, ce qui sera le cas la plupart du temps dans ce manuscrit, leur dimension est calculable directement par une formule du formalisme IFS.

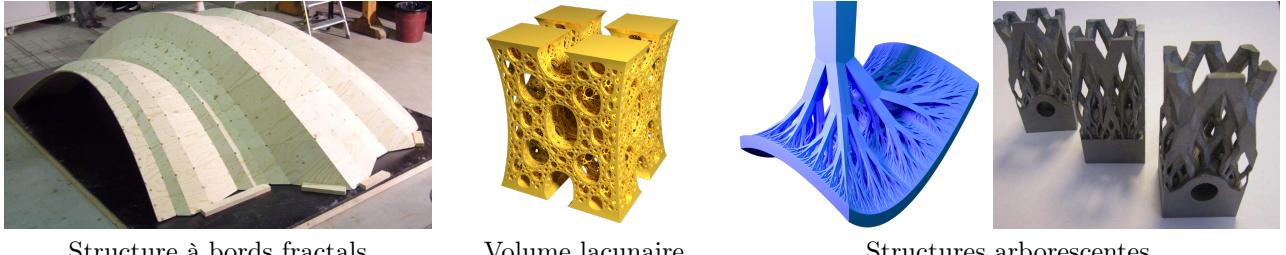


FIGURE 2.2 – Différents exemples d'utilisation de fractales en CGAO. Les replis des structures à bords fractals ont une plus grande résistance à la pression venant du haut que les structures normales car elles répartissent la charge vers l'extérieur. Les volumes lacunaires et les structures arborescentes permettent un gain de matière et une meilleure évacuation de la chaleur sans trop diminuer la résistance de l'objet.

### 2.1.3 Utilisation des fractales en CGAO

Bien qu'elles puissent apporter des solutions à de nombreux problèmes d'optimisation (voir les exemples de la Figure 2.2), les fractales ne sont pas tellement utilisées en CGAO. Pourtant, si les fractales sont si nombreuses à l'état naturel, c'est parce qu'elles sont la réponse obtenue par l'évolution à certains problèmes complexes. Par exemple, les poumons maximisent leur surface extérieure grâce à de nombreux replis pour optimiser les échanges gazeux sans pour autant dépasser le volume autorisé par la cage thoracique. Ce résultat est similaire à un flocon de Koch, qui maximise le périmètre dans une aire donnée, étendu en 3D. De même, la structure interne des os est poreuse et orientée, créant ainsi un volume lacunaire qui est plus léger (car poreux) mais de résistance similaire dans la direction de la longueur de l'os (car orienté).

Malgré tout, les fractales sont encore trop peu utilisées en CGAO. Ceci vient probablement du fait qu'elles soient faciles à générer informatiquement mais beaucoup plus difficile à contrôler et à fabriquer avec les méthodes industrielles traditionnelles. Avec l'évolution des imprimantes 3D, qui permettent de plus facilement générer des objets de structure complexe, le problème sera peut-être résolu dans les années futures et des nombreuses fractales, construites par fabrication additive, apparaîtront.

## 2.2 Systèmes Itérés de Fonctions (IFS)

### 2.2.1 Crédit à Hutchinson

Les IFS sont une notion introduite par Hutchinson [Hut81] pour décrire les fractales en se basant sur l'auto-similarité de celles-ci. En effet, comme chaque sous-partie d'une fractale est elle-même une fractale identique, toute fractale peut être définie par un ensemble de transformations contractantes  $\{\mathcal{T}_0 \dots \mathcal{T}_{n-1}\}$ . Une transformation  $\mathcal{T}_i$  d'un espace métrique complet  $E$  est dite contractante si pour toute paire de points distincts  $A$  et  $B$  de l'espace, la distance entre les deux points est réduite par l'application de la transformation :

$$\text{dist}(A; B) > \text{dist}(\mathcal{T}_i(A); \mathcal{T}_i(B))$$

D'après le **Théorème du point fixe de Banach** [Ban22], toute transformation contractante converge par itération successive vers un point  $\mathcal{P}_i$  unique, appelé point-fixe de la contraction, défini par l'équation suivante :

$$\mathcal{T}_i(\mathcal{P}_i) = \mathcal{P}_i$$

Grâce à la convergence, le point-fixe peut être calculé par une infinité d'applications successives de la transformation à n'importe quel point  $p \in E$  :

$$\mathcal{P}_i = \underbrace{\mathcal{T}_i \circ \dots \circ \mathcal{T}_i}_{\infty}(p)$$

A partir d'un ensemble de transformations contractantes, Hutchinson [Hut81] définit un opérateur  $\mathcal{H}$  (appelé **opérateur de Hutchinson**) tel que pour tout compact  $\mathcal{K}$  :

$$\mathcal{H}(\mathcal{K}) = \bigcup_{i=0}^{n-1} \mathcal{T}_i(\mathcal{K})$$

Les propriétés des transformations contractantes peuvent être transposées à l'opérateur de Hutchinson qui est lui-même contractant. En effet, il existe un unique compact  $\mathcal{A}$ , appelé attracteur de l'IFS, qui est solution de l'équation suivante :

$$\mathcal{H}(\mathcal{A}) = \mathcal{A}$$

Et de la même manière que précédemment, l'attracteur peut être calculé grâce à la propriété de contraction de l'opérateur de Hutchinson :

$$\mathcal{A} = \underbrace{\mathcal{H} \circ \dots \circ \mathcal{H}}_{\infty}(\mathcal{K})$$

Par ailleurs, l'attracteur de toute sous-partie d'un IFS (*i.e.* un IFS n'ayant qu'une partie des transformations de l'IFS parent) est inclus dans l'attracteur de l'IFS global. Par exemple, les points fixes de chaque transformation appartiennent à l'attracteur :

$$\forall i \in \{0 \dots (n-1)\}, \mathcal{P}_i \subset \mathcal{A}$$

Un exemple courant d'IFS est celui du triangle de Sierpiński qui comporte trois transformations qui sont toutes des homothéties de rapport 0.5 et dont les centres de transformations définissent les trois sommets du triangle de départ (voir Figure 2.3).

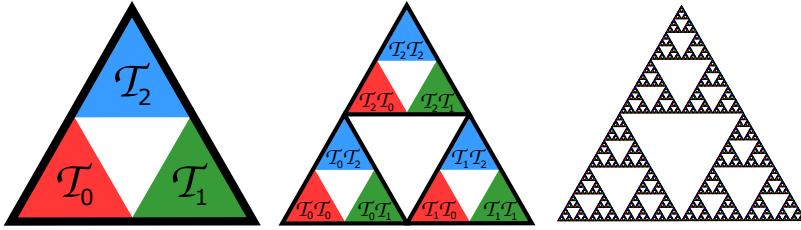


FIGURE 2.3 – IFS du triangle de Sierpinsky. De gauche à droite, le résultat d'une itération de l'opérateur de Hutchinson, d'une deuxième itération, et une représentation de la surface limite.

Les attracteurs d'IFS peuvent être paramétrés par des mots infinis appelés **adresses**. L'IFS étant constitué de transformations contractantes, une composition finie quelconque de ces transformations, va transformer l'attracteur en une de ses parties (voir figure 2.3). Ainsi chaque partie de l'attracteur peut être identifiée par une suite finie de transformations. Plus la longueur de cette suite est importante, plus la partie est petite. Par extension, une suite infinie correspond un point de l'attracteur.

**Définition.** *Etant donné un IFS  $\{\mathcal{T}_i\}_{i \in \{0, \dots, n-1\}}$ , pour tout point  $p$  de l'attracteur, il existe une suite infinie  $\{\sigma_k\}_{k \in \mathbb{N}}$  (avec  $\sigma_k \in \{0, \dots, n-1\}$ ) telle que :*

$$\lim_{\ell \rightarrow \infty} \mathcal{T}_{\sigma_0} \circ \dots \circ \mathcal{T}_{\sigma_\ell}(q) = p, \quad \text{où } q \text{ est un point quelconque de } E.$$

- La suite  $\{\sigma_k\}_{k \in \mathbb{N}}$  est appelé **adresse** de  $p$ .
- La fonction, notée  $\phi$  qui à une adresse associe le point correspondant est appelée **fonction d'adressage** :

$$\phi(\sigma) = \lim_{\ell \rightarrow \infty} \mathcal{T}_{\sigma_0} \circ \dots \circ \mathcal{T}_{\sigma_\ell}(q), \quad \text{où } q \text{ est un point quelconque de } E.$$

Classiquement les adresses sont définies comme une suite d'indices mais dans la suite elles sont notées comme une suite de noms de transformation. Des adresses spécifiques sont utilisées pour calculer une tessellation des courbes et des surfaces. Ces adresses sont composées de deux parties : une partie de longueur finie quelconque et une partie contenant la répétition infinie du même terme  $\mathcal{T}_{\sigma_0} \dots \mathcal{T}_{\sigma_l} \mathcal{T}_i^\infty$ . La deuxième partie correspond au point fixe  $\mathcal{P}_i$  de  $\mathcal{T}_i$ . L'adresse  $\mathcal{T}_{\sigma_0} \dots \mathcal{T}_{\sigma_l} \mathcal{T}_i^\infty$  est alors écrite  $\mathcal{T}_{\sigma_0} \dots \mathcal{T}_{\sigma_l} \mathcal{P}_i$ . Le calcul du point d'adresse  $\mathcal{T}_{\sigma_0} \dots \mathcal{T}_{\sigma_l} \mathcal{P}_i$  est immédiat :  $\phi(\mathcal{T}_{\sigma_0} \dots \mathcal{T}_{\sigma_l} \mathcal{P}_i) = \mathcal{T}_{\sigma_0} \circ \dots \circ \mathcal{T}_{\sigma_l}(\mathcal{P}_i)$  et correspond à un point de l'attracteur. Le point fixe  $\mathcal{P}_i$  peut être calculé par analyse des vecteurs propres dans le cas d'applications linéaires.

### 2.2.2 Développement par Barnsley

Par la suite, Barnsley développe les IFS dans son livre *Fractals Everywhere* [Bar14] (publication originale en 1988, la citation correspond à la réédition de 2014). En particulier, Barnsley démontre le **Théorème du collage** : "tout ensemble compact de points de l'espace euclidien peut être approché par l'attracteur d'un IFS" et propose une méthode itérative qui permet de trouver un IFS approximant un ensemble compact donné. Cette technique, appelée "compression fractale", est utilisée dans le domaine de compression d'images 2D.

En 1993, Barnsley propose une méthode appelée **Jeu du Chaos** permettant d'afficher de manière efficace un IFS sur ordinateur. Cette méthode se base sur le fait que l'image d'un point de l'attracteur par n'importe quelle transformation de l'IFS appartient elle aussi à l'attracteur. De plus, tout point de l'espace converge vers un des points de l'attracteur par itération successive de transformations de l'IFS sur ce point.

À partir d'un point quelconque de l'espace, une transformation de l'IFS est choisie aléatoirement et appliquée au point. Après un certain nombre d'itérations du processus, le point est considéré comme appartenant à l'attracteur et est affiché. À partir de cet instant, chaque itération va afficher un nouveau point jusqu'à ce que suffisamment de points soient affichés. Un exemple appelé Fougère de Barnsley est donné en [Figure 2.4](#) dont les transformations utilisées pour la générer sont les suivantes :

$$\begin{aligned}\mathcal{T}_0 : \begin{pmatrix} x \\ y \end{pmatrix} &\mapsto \begin{pmatrix} 0 \\ 0.16y \end{pmatrix} \\ \mathcal{T}_1 : \begin{pmatrix} x \\ y \end{pmatrix} &\mapsto \begin{pmatrix} 0.85x + 0.04y \\ -0.04x + 0.85y + 1.6 \end{pmatrix} \\ \mathcal{T}_2 : \begin{pmatrix} x \\ y \end{pmatrix} &\mapsto \begin{pmatrix} 0.2x - 0.26y \\ 0.23x + 0.22y + 1.6 \end{pmatrix} \\ \mathcal{T}_3 : \begin{pmatrix} x \\ y \end{pmatrix} &\mapsto \begin{pmatrix} -0.15x + 0.28y \\ 0.26x + 0.24y + 0.44 \end{pmatrix}\end{aligned}$$

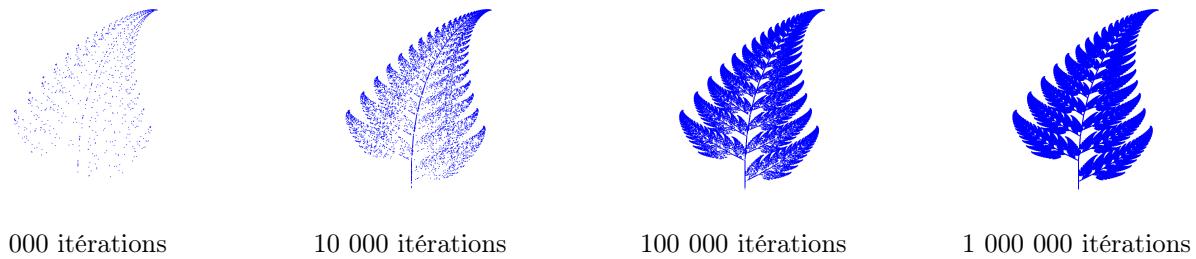


FIGURE 2.4 – Exemple de la génération d'une Fougère de Barnsley par le Jeu du Chaos

Une variante optimisée du Jeu du Chaos consiste à choisir comme point de départ le point-fixe d'une des transformations. Ainsi, le jeu démarre depuis un point de l'attracteur et peut donc directement commencer l'affichage. Pour cela, il faut être capable de calculer de manière efficace le point fixe d'une transformation.

### 2.2.3 Calcul direct des points fixes

Le calcul des points-fixes des transformations se base sur la notion de valeurs, de vecteurs et d'espace propres. Cette notion est d'abord rappelée, puis exploitée pour calculer le point-fixe d'une transformation contractante.

#### Rappel sur les valeurs et vecteurs propres

**Définition.** Soient  $\mathcal{M}$  la matrice carrée de taille  $N \times N$  associée à un endomorphisme (i.e. une application linéaire d'un espace dans lui-même). Les **vecteurs propres**  $\vec{V}_i$  et les **vecteurs propres à gauche**  $\vec{L}_i$  ( $i \in \{1 \dots N\}$ ) correspondent aux directions dans lesquelles l'application se comporte comme une homothétie de rapport  $\lambda_i$  appelé **valeur propre** associée à  $\vec{V}_i$ .

$$\begin{aligned}\mathcal{M} \cdot \vec{V}_i &= \lambda_i \cdot \vec{V}_i \\ \vec{L}_i \cdot \mathcal{M} &= \lambda_i \cdot \vec{L}_i\end{aligned}$$

Pour définir l'ensemble des couples valeurs/vecteurs propres d'une matrice  $\mathcal{M}$ , la méthode est la suivante : l'ensemble des valeurs propres est d'abord calculé, puis pour chaque valeur propre, les vecteurs propres associés. L'ensemble des valeurs propres  $\lambda$  de la matrice  $M$  est l'ensemble des racines du **polynôme caractéristique** :

$$\det(\mathcal{M} - \lambda Id) = 0$$

**Démonstration.** Soient  $\lambda_i$  une valeur propre de  $\mathcal{M}$  et  $\vec{V}_i \neq \vec{0}$  un des vecteurs propres associés.

$$\begin{aligned}\mathcal{M} \cdot \vec{V}_i &= \lambda_i \cdot \vec{V}_i \\ \mathcal{M} \cdot \vec{V}_i - \lambda_i \cdot \vec{V}_i &= \vec{0} \\ \mathcal{M} \cdot \vec{V}_i - \lambda_i Id \cdot \vec{V}_i &= \vec{0} \\ (\mathcal{M} - \lambda_i Id) \cdot \vec{V}_i &= \vec{0} \\ \implies \det(\mathcal{M} - \lambda_i Id) &= 0\end{aligned}$$

□

L'ensemble des valeurs propres ainsi calculé est habituellement ordonné de la valeur dont le module est le plus grand à celle dont le module est le plus proche de 0 :

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_N| > 0$$

Ensuite, pour chaque  $\lambda_i$ , un vecteur propre  $\vec{V}_i$  et un vecteur propre à gauche  $\vec{L}_i$  sont calculés en résolvant les systèmes :

$$\begin{aligned}\mathcal{M} \cdot \vec{V}_i &= \lambda_i \cdot \vec{V}_i \\ \vec{L}_i \cdot \mathcal{M} &= \lambda_i \cdot \vec{L}_i\end{aligned}$$

Ces vecteurs propres sont des représentants de l'ensemble des vecteurs propres car si  $\vec{V}_i$  est une solution, alors  $k\vec{V}_i, k \in \mathbb{R}^*$  l'est aussi. Ceci est également valable pour les vecteurs propres à gauche  $\vec{L}_i$ . Dans les deux cas, la démonstration est triviale.

### Cas des matrices de coordonnées homogènes

Pour pouvoir définir toutes les transformations affines sous forme de matrices, les **coordonnées homogènes** (ou projectives) sont habituellement utilisées en informatique graphique. Tout **point** de l'espace géométrique  $P = (\alpha_1 ; \dots ; \alpha_N) \in \mathbb{R}^d$  se voit attribuer, dans l'**espace projectif**, une nouvelle coordonnée homogène égale à 1 ( $\hat{P} = (\alpha_1 ; \dots ; \alpha_N ; 1)$ ). Sont définies des classes d'équivalence  $\hat{P}_{\omega \neq 0} = (\omega\alpha_1 ; \dots ; \omega\alpha_N ; \omega)$  dont tous les points correspondent, dans l'espace géométrique, à  $P$ . Les **vecteurs** de l'espace géométrique sont représentés dans l'espace projectif par des points dont la coordonnée homogène est égale à 0. La matrice (dite **homogène**) représentant une transformation affine de l'espace projectif est de taille  $(N+1) \times (N+1)$  et a pour dernière ligne  $(0 \dots 0 \ 1)$ .

Pour déterminer si la transformation est contractante ou non, les valeurs propres, puis les vecteurs propres de la matrice sont étudiés.

**Propriété :** Une matrice homogène d'une transformation affine possède au moins une valeur propre égale à 1.

**Démonstration.** Soit une matrice  $\mathcal{M}$  de taille  $(N+1) \times (N+1)$ ,  $m$  sa sous-matrice haut-gauche de taille  $N \times N$  et  $T$  un vecteur colonne  $(N \times 1)$  (représentant une translation). Le polynôme caractéristique de  $\mathcal{M}$  est de la forme :

$$\begin{aligned}\det(\mathcal{M} - \lambda Id) &= \det \left( \begin{pmatrix} m & T \\ \mathbf{0} & 1 \end{pmatrix} - \begin{pmatrix} \lambda Id & \mathbf{0} \\ \mathbf{0} & \lambda \end{pmatrix} \right) \\ &= \det \begin{pmatrix} m - \lambda Id & T \\ \mathbf{0} & 1 - \lambda \end{pmatrix} \\ &= \pm(1 - \lambda) \det(m - \lambda Id)\end{aligned}$$

Donc au moins une des racines du polynôme caractéristique est égale à 1, ce qui implique qu'au moins une des valeurs propres de  $\mathcal{M}$  est égale à 1.  $\square$

La forme du polynôme caractéristique implique également que les  $N$  valeurs propres restantes sont les valeurs propres de la sous-matrice  $m$  et les vecteurs propres associés à ces valeurs propres sont les vecteurs propres de  $m$ . Les vecteurs propres de  $\mathcal{M}$  sont déduits des vecteurs propres de  $m$  en ajoutant une dernière composante qui s'impose à 0. Du fait de la dernière composante nulle, ces vecteurs propres correspondent à des vecteurs de l'espace géométrique.

En résumé : la matrice  $\mathcal{M}$  a **un vecteur propre** associé à la valeur propre 1 qui correspond à **un vecteur de l'espace projectif** et  $N$  **vecteurs propres** qui sont des **vecteurs de l'espace géométrique**.

### Cas des transformations contractantes

Les valeurs propres d'une matrice sont des coefficients de dilatation. Ainsi, pour qu'une matrice corresponde à une transformation contractante, il faut que toutes ses valeurs propres appartiennent à l'intervalle  $]-1; 1[$ . Dans le cas présent, il s'agit d'une contraction de l'espace géométrique représentée par une matrice de coordonnées homogènes. En repartant du résultat précédent, la matrice homogène a un vecteur propre associé à la valeur 1 pour l'espace projectif et  $N$  vecteurs propres pour l'espace géométrique. Comme ce qui est recherché est une contraction de l'espace géométrique, c'est uniquement ces  $N$  derniers vecteurs propres qui doivent correspondre à une valeur propre comprises entre -1 et 1 exclus. Ainsi, les valeurs propres d'une matrice homogène représentant une transformation contractante de l'espace géométrique sont forcément de la forme :

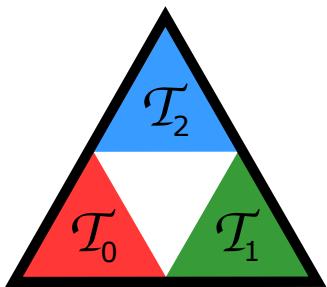
$$\lambda_1 = 1 > |\lambda_2| \dots \geq |\lambda_{(N+1)}| > 0$$

où  $\lambda_1$  est associé à un vecteur de l'espace projectif et  $\lambda_{\{2\dots(N+1)\}}$  à des vecteurs de l'espace géométrique. De cette manière, la transformation contracte l'espace projectif selon l'hyper-plan qu'est l'espace géométrique et est invariant selon l'axe des coordonnées homogènes  $\omega$ . Imaginez un cylindre dont l'axe principal est celui des  $\omega$  et dont le rayon diminue jusqu'à n'être plus qu'une droite, le **point-fixe** de la transformation est l'intersection de cette droite avec l'hyper-plan de l'espace géométrique  $\omega = 1$ .

Pour calculer ce point-fixe qui est le vecteur propre  $\vec{V}_1 = (\alpha_1 ; \dots ; \alpha_N ; \omega)$  associé à  $\lambda_1$  est étudié. Si  $\omega \neq 0$ , alors  $\vec{V}_1$  est un représentant de la classe des vecteurs de l'**espace projectif**  $k.\vec{V}_1 = (k\alpha_1 ; \dots ; k\alpha_N ; k\omega)$ ,  $k \in \mathbb{R}^*$  qui correspondent tous au point de l'**espace géométrique**  $\mathcal{P}_-(\alpha_1/\omega \dots \alpha_N/\omega, 1)$ . Ce point est un vecteur propre associé à la valeur propre 1 (dans l'espace projectif), il n'est donc pas modifié par la transformation : c'est le point fixe unique de la transformation dans l'espace géométrique.

### Exemple du Triangle de Sierpinski

Soient  $\mathcal{T}_0$ ,  $\mathcal{T}_1$ , et  $\mathcal{T}_2$  les transformations du Triangle de Sierpinski (des homothéties de rapport 1/2 dont les centres sont les sommets du triangle équilatéral unitaire) et  $\mathcal{M}_0$ ,  $\mathcal{M}_1$ , et  $\mathcal{M}_2$  les matrices de transformations en coordonnées homogènes associées (voir [Figure 2.5](#)).



$$\mathcal{M}_0 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} ; \quad \mathcal{M}_1 = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} ; \quad \mathcal{M}_2 = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{\sqrt{3}}{4} \\ 0 & 0 & 1 \end{pmatrix}$$

FIGURE 2.5 – Exemple du Triangle de Sierpinski. A gauche la représentation graphique des transformations  $\mathcal{T}_0$ ,  $\mathcal{T}_1$ , et  $\mathcal{T}_2$  où le triangle noir est associé respectivement au sous-triangle rouge, vert, et cyan. A droite, les trois matrices  $\mathcal{M}_0$ ,  $\mathcal{M}_1$ , et  $\mathcal{M}_2$  associées aux transformations.

Les valeurs propres sont pour toutes les matrices de transformations  $\lambda_1 = 1 > \lambda_2 = \lambda_3 = 0.5$  ce qui correspond à la forme que les valeurs propres doivent avoir pour que les transformations soient des contractions. Les vecteurs propres associés aux deux valeurs propres les plus petites sont les vecteurs  $(1 ; 0 ; 0)^\top$  et  $(0 ; 1 ; 0)^\top$ . Ceci confirme que toutes les transformations correspondent à des contractions de rapport  $1/2$  selon les deux axes principaux du plan. Pour les vecteurs propres associés à  $\lambda_1 = 1$ , qui sont donc les points fixes, ils sont respectivement  $\mathcal{P}_0 = (0 ; 0 ; 1)^\top$ ,  $\mathcal{P}_1 = (1 ; 0 ; 1)^\top$ , et  $\mathcal{P}_2 = (1/2 ; \sqrt{3}/2 ; 1)^\top$  ce qui correspond aux centres des trois homothéties.

### 2.2.4 Les produits tensoriels

Une méthode habituelle de génération d'une surface est le produit-tensoriel de deux courbes. Pour générer ces surfaces, il suffit donc de définir le produit-tensoriel d'IFS. Soient  $\mathbf{A}$  et  $\mathbf{B}$  deux IFS dont les transformations sont respectivement  $\mathcal{T}_{\mathbf{A},0} \dots \mathcal{T}_{\mathbf{A},m-1}$  et  $\mathcal{T}_{\mathbf{B},0} \dots \mathcal{T}_{\mathbf{B},n-1}$ . Le nombre de transformations de l'IFS  $\mathbf{A} \otimes \mathbf{B}$  est  $m \times n$  et ces transformations sont toutes les combinaisons possibles de  $\mathcal{T}_{\mathbf{A},i}$  et  $\mathcal{T}_{\mathbf{B},j}$ . La matrice associée à une combinaison de transformation est le produit de Kronecker des deux matrices des transformations de départ :  $\mathcal{M}_{i,j} = \mathcal{M}_{\mathbf{A},i} \otimes \mathcal{M}_{\mathbf{B},j}$

Par exemple, un ensemble de Cantor est défini par les matrices  $\mathcal{M}_{\mathcal{L}} = \begin{pmatrix} 1 & 0 \\ 2/3 & 1/3 \end{pmatrix}$  et  $\mathcal{M}_{\mathcal{R}} = \begin{pmatrix} 1/3 & 2/3 \\ 0 & 1 \end{pmatrix}$ .

Son produit tensoriel est défini par les quatres transformations et les quatre matrices associées sont données dans la Figure 2.6.

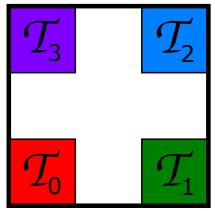
	$\mathcal{M}_{\mathcal{R}}$	$\mathcal{M}_3 = \begin{pmatrix} 1/3 & 0 & 0 & 2/3 \\ 2/9 & 1/9 & 2/9 & 4/9 \\ 0 & 0 & 1/3 & 2/3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\mathcal{M}_2 = \begin{pmatrix} 1/9 & 2/9 & 4/9 & 1/9 \\ 0 & 1/3 & 2/3 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2/3 & 1/3 \end{pmatrix}$
$\mathcal{M}_{\mathcal{L}}$	$\mathcal{M}_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2/3 & 1/3 & 0 & 0 \\ 4/9 & 2/9 & 1/9 & 2/9 \\ 2/3 & 0 & 0 & 1/3 \end{pmatrix}$	$\mathcal{M}_1 = \begin{pmatrix} 1/3 & 2/3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2/3 & 1/3 & 0 \\ 2/9 & 4/9 & 2/9 & 1/9 \end{pmatrix}$	$\mathcal{M}_{\mathcal{L}}$
$\otimes$	$\mathcal{M}_{\mathcal{L}}$	$\mathcal{M}_{\mathcal{R}}$	

FIGURE 2.6 – IFS correspondant au produit-tensoriel de deux ensembles de Cantor. A gauche, les transformations géométriques qui associent le carré unitaire à un carré inclus dans le carré unitaire. A droite les matrices de transformations calculées par produit de Kronecker. Attention, comme les points sont notés dans l'ordre trigonométrique pour conserver la cohérence avec le reste du manuscrit, les 3ème et 4ème colonnes/lignes sont échangées par rapport au résultat obtenu par produit de Kronecker.

### 2.2.5 Calcul de la dimension fractale d'un IFS

Le calcul de la dimension de Hausdorff d'une fractale peut être très fastidieux. Dans le cas d'une fractale représentée par un IFS dont les transformations sont disjointes (*i.e.*  $\forall(i, j), \mathcal{T}_i(\mathcal{F}) \cap \mathcal{T}_j(\mathcal{F}) = \emptyset$ ), ce calcul peut être simplifié. En effet, dans le cas particulier où les  $n$  transformations de l'IFS sont contractantes de rapport  $k$ , la dimension fractale  $\mathbf{d}$  de l'attracteur est :

$$\mathbf{d} = \frac{\ln(n)}{\ln(1/k)} = \log_{1/k}(n)$$

Par exemple, l'ensemble de Cantor est composé de deux homothéties de rapport  $1/3$ , sa dimension fractale est donc  $\frac{\ln(2)}{\ln(3)}$ . Les dimensions de quelques fractales classiques sont résumées dans le tableau suivant :

Nom	n	k	d
Ensemble de Cantor	2	$1/3$	$\frac{\ln(2)}{\ln(3)} = \log_3(2) \approx 0.63$
Flocon de Koch	4	$1/3$	$\frac{\ln(4)}{\ln(3)} = \log_3(4) \approx 1.26$
Triangle de Sierpinski	3	$1/2$	$\frac{\ln(3)}{\ln(2)} = \log_2(3) \approx 1.58$
Tapis de Sierpinski	8	$1/3$	$\frac{\ln(8)}{\ln(3)} = \log_3(8) \approx 1.89$

Dans le cas où les  $n$  transformations ont des rapports de contractions  $k_i$  différents, la dimension de Hausdorff est la solution de l'équation suivante :

$$\sum_{i=1}^n k_i^d = 1$$

Par exemple, l'ensemble de Cantor asymétrique est composé de deux transformations de rapport  $1/2$  et  $1/4$ , sa dimension de Hausdorff est donc :

$$\left(\frac{1}{2}\right)^d + \left(\frac{1}{4}\right)^d = 1 \Leftrightarrow d = \varphi = \frac{1 + \sqrt{5}}{2} \text{ (le nombre d'or)}$$

## 2.3 Les IFS barycentriques

En CGAO, les courbes et surfaces sont habituellement définies par un ensemble ordonné de points de contrôle (*e.g.* polygone, maillage) qui permet des modifications intuitives de la forme de la surface pour l'utilisateur. Cette méthode est appelée *free-form modeling*. Les IFS n'ont pas de notion d'ensemble de contrôle, ils sont définis uniquement par des transformations de l'espace géométrique : pour modifier la forme de l'attracteur, il faut modifier les transformations ce qui est moins intuitif que la notion de points de contrôle.

Goldman [Gol04] pour les courbes de Bézier, puis Schaeffer, Levin et Goldman [SLG05] pour les B-Splines uniformes et les fractales classiques (*e.g.* le Triangle de Sierpiński et Flocon de Koch) montrent qu'il est possible de faire du *free-form modeling* sur des IFS. Chaque transformation de l'IFS associe un ensemble de contrôle (polygone ou maillage) à un nouvel ensemble de contrôle. La forme de l'attracteur dépend alors de la position des points de contrôle et donc peut être modifiée par l'utilisateur en agissant uniquement sur l'ensemble de contrôle original. Dans ces travaux, l'attracteur est directement calculé dans l'espace géométrique, ce qui implique que toute modification de l'ensemble de contrôle entraîne un recalcul de l'attracteur ce qui est contraignant, en particulier lorsque celui-ci est conséquent. Pourtant, dans des travaux antérieurs rarement cités dans la littérature, Zaïr et Tosan [ZT96] introduisent la notion d'**IFS projeté** qui fait intervenir deux espaces différents : un espace d'itération dans lequel l'IFS est construit et un espace de modélisation dans lequel il est projeté. Ainsi, l'attracteur est calculé une fois pour toutes dans l'espace itératif, et c'est uniquement la projection, moins coûteuse, qui doit être recalculée en cas de modification de l'ensemble de contrôle.

Dans le cas présent, nous parlerons d'**IFS barycentriques** qui correspondent aux IFS projetés introduits dans [ZT96]. De façon analogue aux courbes et surfaces à pôles, l'attracteur (correspondant aux fonctions de bases) est construit dans un espace barycentrique puis projeté dans l'espace de modélisation à partir d'un ensemble de points de contrôle. Parallèlement à cet IFS définissant la géométrie de l'objet, est défini un second IFS *compatible* définissant l'espace des paramètres. Prenons l'exemple, d'une surface quadrangulaire se subdivisant en quatre sous-partie. Cette surface sera définie dans l'espace barycentrique comme attracteur d'un IFS composé de quatre transformations. L'espace des paramètres est alors défini par un deuxième  $IFS_p$  composé également de quatre transformations et dont l'attracteur est  $[0, 1]^2$ . Les adresses permettent de mettre en correspondance les points de l'espace des paramètres et ceux de la surface pour obtenir une expression paramétrique standard de la surface :

$$S(u, v) = \phi(\phi_p^{-1}((u, v))), \forall (u, v) \in [0, 1]^2,$$

où  $\phi$  représente la fonction d'adressage de l'IFS de la surface, et  $\phi_p$  celle de l'IFS de l'espace des paramètres. Le calcul de  $\phi_p^{-1}$  revient à déterminer l'adresse correspondante aux points  $(u, v) \in [0, 1]^2$ . Intuitivement, la compatibilité entre les IFS exprime le fait que si pour l'un des attracteurs deux sous-parties s'intersectent, la même intersection doit être observée sur l'autre attracteur. Plus précisément, les attracteurs doivent avoir les mêmes ensemble d'adresses multiples. La tesselation de la surface peut alors être construite à partir d'une tesselation de l'espace des paramètres et l'expression paramétrique de la surface.

### 2.3.1 Génération des adresses dans l'espace des paramètres

Pour transporter la tesselation de l'espace des paramètres vers la surface, il faut calculer  $\phi_p^{-1}$ , c'est-à-dire déterminer l'adresse d'un point de l'espace des paramètres. Cette adresse peut éventuellement être multiple mais en raison de la contrainte de compatibilité entre les IFS (celui de l'objet et celui de l'espace des paramètres) il suffit d'en trouver une seule.

Dans le cas d'une courbe paramétrique, tout point de la courbe est défini par un paramètre  $t \in [0; 1]$  et est noté  $\mathcal{C}(t)$ . L'espace des paramètres est donc  $[0; 1]$  et toute transformation de lIFS, définissant l'espace des paramètres, associe  $[0; 1]$  à  $[a; b]$  où  $0 \leq a \leq b \leq 1$ . Pour que la transformation converge vers un point fixe unique de l'espace des paramètres,  $[a; b]$  doit être différent de  $[0; 1]$ . Une fois les transformations et les points fixes associés définis, l'adresse est générée de manière itérative :

- si  $t$  correspond à un point fixe, le  $\mathcal{P}_i$  correspondant est ajoutée à l'adresse et la génération est terminée
- sinon la transformation  $\mathcal{T}_i$  qui correspond à l'intervalle auquel appartient  $t$  est ajoutée à l'adresse et la méthode est recommandée pour  $t' = \mathcal{T}_i^{-1}(t)$
- si aucune transformation ne correspond, c'est que le point de paramètre  $t$  n'appartient pas à l'attracteur

Par exemple, considérons le cas le plus courant où les deux transformations et les points fixes associés de lIFS sont définies comme ceci dans l'espace paramétrique :

$$\begin{aligned}\mathcal{T}_0(x) &= \frac{1}{2}x, \quad \forall x \in \mathbb{R} \\ \mathcal{T}_1(x) &= \frac{1}{2}x + \frac{1}{2}, \quad \forall x \in \mathbb{R} \\ \mathcal{P}_0(x) &= 0, \quad \forall x \in \mathbb{R} \\ \mathcal{P}_1(x) &= 1, \quad \forall x \in \mathbb{R}\end{aligned}$$

Tout paramètre  $t$  diadique (*i.e.* de la forme  $\{0 \dots 2^{l-1}\}/2^{l-1}$ ) possède une adresse de longueur maximale  $l$ . Par exemple, l'adresse du paramètre  $t = 0.75$  est définie comme ceci :

- $0.75 \in [0.5; 1]$  donc  $\mathcal{T}_1$  est appliquée et  $t \mapsto \mathcal{T}_1^{-1}(t) = 2t - 1 = 0.5$
- $0.5 \in [0; 0.5]$  donc  $\mathcal{T}_0$  est appliquée et  $t \mapsto \mathcal{T}_0^{-1}(t) = 2t = 1$
- 1 correspond à  $\mathcal{P}_1$

L'adresse de  $t = 0.75$  est donc  $\mathcal{T}_1\mathcal{T}_0\mathcal{P}_1$ . Mais cette adresse n'est pas unique : l'adresse  $\mathcal{T}_1\mathcal{T}_1\mathcal{P}_0$  est également valable. Dans certains cas la multiplicité des adresses est importante (par exemple lorsque la courbe est disjointe en ce paramètre ou quand il y a une discontinuité de tangente au niveau d'un point fixe) et les différentes adresses sont conservées alors que dans d'autres elle n'a pas d'influence et l'unicité est privilégiée en donnant un ordre de priorité aux transformations.

Toute tesselation de l'attracteur peut être définie comme une liste d'adresses. Par exemple, une tesselation de paramètres  $\{0; 0.25; 0.5; 0.75; 1\}$  correspond à la liste d'adresse  $\{\mathcal{P}_0, \mathcal{T}_0\mathcal{T}_0\mathcal{P}_1, \mathcal{T}_0\mathcal{P}_1, \mathcal{T}_1\mathcal{T}_1\mathcal{P}_0, \mathcal{P}_1\}$  Cette liste est la base de la génération de la même tesselation de l'attracteur, mais cette fois dans l'espace barycentrique.

Dans le cas d'une surface, l'espace des paramètres est appelé par assimilation espace des textures et est défini comme  $[0; 1] \times [0; 1]$ . Les points de la surface sont notés  $\mathcal{S}(u; v)$  et les adresses sont générées de la même manière que précédemment, la seule différence est que le choix des transformations se fait en prenant en compte deux paramètres.

Attention, dans certains cas, tous les paramètres de  $[0; 1]$  ne correspondent pas à une adresse. Par exemple, pour un ensemble de Cantor, les transformations sont les suivantes :

$$\begin{aligned}\mathcal{T}_0(x) &= \frac{1}{3}x, \quad \forall x \in \mathbb{R} \\ \mathcal{T}_1(x) &= \frac{1}{3}x + \frac{2}{3}, \quad \forall x \in \mathbb{R} \\ \mathcal{P}_0(x) &= 0, \quad \forall x \in \mathbb{R} \\ \mathcal{P}_1(x) &= 1, \quad \forall x \in \mathbb{R}\end{aligned}$$

et donc le point  $t = 1/2$  qui n'appartient pas à l'attracteur ne peut pas être défini par une adresse. Ceci doit être gardé à l'esprit lorsque les paramètres de la tesselation sont choisis.

### 2.3.2 Génération d'une tessellation de l'attracteur dans l'espace barycentrique $\mathbf{BI}^N$

Cette sous-section est organisée de la manière suivante. Premièrement l'espace barycentrique et ses propriétés sont définies. Ensuite les conditions nécessaires et suffisantes à l'existence de transformations contactantes sont démontrées, et la méthode pour calculer directement les points-fixes de ces transformations est décrite. Pour finir, une tessellation de l'attracteur est construite et la liste des combinaisons barycentriques est déduite.

#### L'espace barycentrique $\mathbf{BI}^N$ et ses propriétés

**Définition.** L'espace barycentrique  $\mathbf{BI}^N$  est un sous-espace de  $\mathbb{R}^N$ . Plus précisément, il s'agit de l'hyper-plan composé de l'ensemble des points dont la somme des coordonnées est égale à 1. Tous les points de  $\mathbf{BI}^N$  sont les coefficients d'une combinaison barycentrique.

$$P(\alpha_1; \dots; \alpha_N) \in \mathbf{BI}^N \Leftrightarrow \sum_{i=1}^N \alpha_i = 1$$

Une représentation visuelle de  $BI^3$  est donnée en Figure 2.7

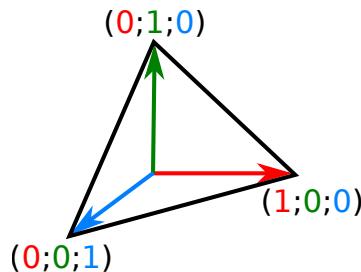


FIGURE 2.7 – Représentation schématique de l'espace barycentrique de dimension 3,  $\mathbf{BI}^3$ . Les vecteurs rouge, vert, et cyan sont la base orthonormale de  $\mathbb{R}^3$  et le triangle noir définit la base d'un plan infini qui correspond à  $\mathbf{BI}^3$ .

Pour respecter la cohérence de cet espace, les transformations doivent respecter des règles spécifiques. Par exemple, une translation se fait par un vecteur dont la somme des coordonnées est nulle (une translation qui ne respecterait pas cette condition ferait "sortir" les points de l'hyper-plan). L'espace barycentrique étant un espace affine, tout endomorphisme affine de cet espace correspond à une matrice  $\mathcal{M}$ . Cette matrice est appliquée à droite pour transformer  $P(\alpha_1; \dots; \alpha_N) \in \mathbf{BI}^N$  en  $P'(\alpha'_1; \dots; \alpha'_N) \in \mathbf{BI}^N$  :

$$P' = P \cdot \mathcal{M}$$

Pour s'assurer que le point image appartienne à l'espace barycentrique quel que soit le point antécédent, la transformation doit respecter la propriété suivante :

**Propriété.** Tout endomorphisme de l'espace barycentrique est représenté par une matrice dite barycentrique (i.e. dont la somme de chaque ligne est égale à 1). Les matrices barycentriques sont une généralisation des matrices stochastiques (parfois dites de Markov) où les coefficients peuvent être négatifs.

### 2.3.3 Calcul direct du point-fixe et de la tangente d'une transformation dans l'espace barycentrique

Toute transformation de l'espace barycentrique doit être décrite sous la forme d'une matrice stochastique mais ce n'est pas suffisant pour que la transformation soit intégrée dans un IFS. En effet, il faut également que la transformation soit contractante (avec les mêmes règles que dans l'espace géométrique).

**Propriété.** Tout endomorphisme contractant de l'espace barycentrique correspond à une matrice barycentrique dont les valeurs propres sont de la forme :

$$\lambda_1 = 1 > |\lambda_2| \dots \geq |\lambda_d| > 0$$

*N.B.* : une matrice barycentrique possède toujours une de ses valeurs propres égale à 1 car le vecteur colonne composé uniquement de 1 n'est pas modifié par l'application d'une matrice barycentrique et donc est un vecteur propre à droite associé à la valeur propre 1. Mais, il faut également qu'aucune autre valeur propre ne soit supérieure à 1 (sinon il y a dilatation) ou égale à 1 (dans ce cas il y aurait au moins deux points fixes).

Soit  $\mathcal{T} : \mathbf{BI}^N \mapsto \mathbf{BI}^N$  une transformation contractante d'un IFS de l'espace barycentrique et  $\mathcal{M}$  la matrice stochastique de transformation associée. Comme la valeur propre principale  $\lambda_1$  est égale à 1, il existe un vecteur ligne  $\vec{L}_1$  tels que :

$$\vec{L}_1 \mathcal{M} = \vec{L}_1$$

$\vec{L}_1$  est un représentant du sous-espace propre associé à  $\lambda_1$ . L'unique représentant de cette classe dont la somme des coordonnées est égale à 1 est le point-fixe  $\mathcal{B}$  de la transformation.

Lorsque l'IFS barycentrique définit une courbe d'un seul paramètre,  $\vec{L}_2$  (le vecteur propre associé à la deuxième plus grande valeur propre) est un vecteur propre dont la somme des coordonnées est nulle, il s'agit donc d'un vecteur de l'espace barycentrique. Sous certaines conditions [Ben09, Pod13],  $\vec{L}_2$  est la tangente de l'attracteur en  $\mathcal{B}$  notée  $\vec{\tau}$ . Si un point a plusieurs adresses, il est possible que les tangentes obtenues par les différentes adresses ne soient pas colinéaires, ce qui veut dire qu'il y a discontinuité de tangente en ce point.

Dans le cas d'un IFS barycentrique définissant une surface (à deux paramètres) les couples valeur/vecteur propres sont pris deux à deux pour le calcul des dérivées. C'est à dire  $\vec{\tau} = \vec{L}_2$  et  $\vec{\tau}' = \vec{L}_3$  sont un couple de vecteurs directeurs du plan tangent. Lorsque cet IFS est construit par produit-tensoriel de deux IFS barycentriques à un seul paramètre, il n'est pas nécessaire de recalculer le point-fixe et les dérivées par analyse de valeurs/vecteurs propres. En effet, ceux-ci peuvent être calculés directement par les propriétés suivantes :

**Propriété.** Soit  $\mathcal{T}$  une transformation composée de deux transformations  $\mathcal{T}_A$  et  $\mathcal{T}_B$ . La matrice de transformation de  $\mathcal{T}$  est construite par  $\mathcal{M} = \mathcal{M}_A \otimes \mathcal{M}_B$  (où  $\otimes$  désigne le produit de Kronecker). Sont alors définis les vecteurs suivants :

$$\begin{aligned}\mathcal{B} &= \mathcal{B}_A \otimes \mathcal{B}_B \\ \vec{\tau} &= \mathcal{B}_A \otimes \vec{\tau}_B \\ \vec{\tau}' &= \vec{\tau}_A \otimes \mathcal{B}_B\end{aligned}$$

**Exemple de l'ensemble de Cantor tensoriel :**

Soient les transformations  $\mathcal{T}_{\mathcal{L}}$  et  $\mathcal{T}_{\mathcal{R}}$  représentées par  $\mathcal{M}_{\mathcal{L}} = \begin{pmatrix} 1 & 0 \\ 2/3 & 1/3 \end{pmatrix}$  et  $\mathcal{M}_{\mathcal{R}} = \begin{pmatrix} 1/3 & 2/3 \\ 0 & 1 \end{pmatrix}$ .

Les valeurs propres des deux matrices sont :  $\lambda_1 = 1$  et  $\lambda_2 = 1/3$ , ce qui indique que chaque transformation à un point fixe et génère une contraction de rapport  $1/3$ . Les points fixes associés sont  $\mathcal{B}_{\mathcal{L}} = (1; 0)$  et  $\mathcal{B}_{\mathcal{R}} = (0; 1)$  et les tangentes aux points fixes sont  $\vec{\tau}_{\mathcal{L}} = (-1; 1)$  et  $\vec{\tau}_{\mathcal{R}} = (1; -1)$ .

Les matrices  $\mathcal{M}_{0\dots 3}$  des transformations de l'ensemble de Cantor tensoriel sont définies par produit de Kronecker dans [Figure 2.6](#). En prenant comme exemple la matrice  $\mathcal{M}_0$ , le point-fixe et les tangentes en ce point sont :

$$\begin{aligned}\mathcal{B}_0 &= \mathcal{B}_{\mathcal{L}} \otimes \mathcal{B}_{\mathcal{L}} = (1; 0; 0; 0) \\ \vec{\tau}_0 &= \mathcal{B}_{\mathcal{L}} \otimes \vec{\tau}_{\mathcal{L}} = (-1; 1; 0; 0) \\ \vec{\tau}'_0 &= \vec{\tau}_{\mathcal{L}} \otimes \mathcal{B}_{\mathcal{L}} = (-1; 0; 0; 1)\end{aligned}$$

Attention, pour respecter l'ordre trigonométrique, les 3ème et 4ème colonnes/lignes des matrices ont été échangées par rapport au "vrai" produit de Kronecker, il en est donc de même sur les 3ème et 4ème coordonnées des vecteurs.

### Construction d'une tessellation de l'attracteur dans l'espace barycentrique

Soit  $\mathcal{M}_i$  la représentation matricielle de chaque transformation  $\mathcal{T}_i$ . Le point-fixe  $\mathcal{B}_i$  de  $\mathcal{T}_i$  est calculé à partir de  $\mathcal{M}_i$ . Les adresses définissant une tessellation de l'espace des paramètres sont utilisées pour construire une tessellation équivalente de l'attracteur dans l'espace barycentrique. Ainsi à chaque adresse  $\mathcal{T}_a \dots \mathcal{T}_y \mathcal{P}_z$  est associé le point de l'attracteur de l'espace barycentrique  $\mathcal{B}_z \mathcal{M}_y \dots \mathcal{M}_a$ . Ainsi, la liste d'adresses, est transformée en une liste de combinaisons barycentriques.

#### 2.3.4 Projection d'une combinaison barycentrique dans l'espace des points de contrôle

L'application d'une combinaison barycentrique sur un ensemble de  $n$  points d'un espace euclidien (classiquement  $\mathbb{R}^3$ ) correspond à une projection de  $\mathbf{BI}^n$  dans l'espace euclidien (ou par abus de langage un projection dans l'ensemble des points de contrôle). Le point ainsi obtenu est la représentation de  $\mathcal{B}$  dans l'espace des points de contrôle. A partir de la liste des combinaisons barycentriques, il suffit de les appliquer sur l'ensemble de contrôle pour projeter la tessellation de l'attracteur de l'espace barycentrique vers l'espace des points de contrôle (voir Figure 2.8).

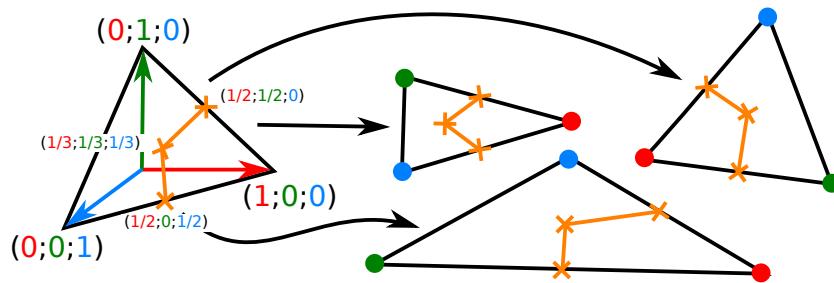


FIGURE 2.8 – Projection de la tessellation "milieu d'arête - centre de gravité du triangle - milieu d'arête" dans différents triangles de contrôle. Le centre de gravité et les milieux d'arêtes sont représentés par les point de coordonnées  $(1/3; 1/3; 1/3)$ ,  $(1/2; 1/2; 0)$ , et  $(1/2; 0; 1/2)$  (les croix oranges) dans l'espace barycentrique. Ces points sont ensuite plongés dans les différents triangle de contrôle.

Pour connaître les propriétés différentielles, la tangente  $\vec{T}_i$  du point projeté est calculée en projetant  $\vec{\tau}_i$  de l'espace barycentrique vers l'espace des points de contrôle. Si plusieurs tangentes sont définies pour le même point, elles sont toutes projetées.

Dans le cas d'une surface générée par produit-tensoriel, les deux tangentes principales  $\vec{T}_i$  et  $\vec{T}'_i$  en un point sont obtenues en plongeant  $\vec{\tau}_i$  et  $\vec{\tau}'_i$ . La normale  $\vec{N}_i$  en ce point est calculée par produit vectoriel de ces deux tangentes. La discontinuité des tangentes entraîne une discontinuité des normales, dans ce cas différentes normales sont calculées comme les différentes combinaisons de tangentes possibles.

#### 2.3.5 Résumé de la méthode

Premièrement, une tessellation de l'attracteur est choisie comme une liste de points de l'espace des paramètres. Dans l'espace des paramètres, cette liste est convertie en liste d'adresses. Cette liste d'adresse est utilisée pour calculer les points de la tessellation de l'attracteur dans l'espace barycentrique. Ces points sont ensuite projetés dans l'espace de modélisation, en appliquant la combinaison barycentrique sur les points de contrôle. Ainsi, une tessellation de l'attracteur de l'IFS correspondant à l'ensemble de contrôle est générée.

#### Exemple de la courbe de Koch

La courbe de Koch est contruite itérativement à partir d'un segment de longueur 1 dont le tiers central est remplacé par deux segments de longueur  $1/3$  en pointe. La méthode est ensuite réappliquée sur les quatre nouveaux segments. Dans cet exemple, l'IFS barycentrique génère la courbe obtenue après deux itérations.

L'attracteur étant une courbe, l'espace des paramètres est de dimension 1. De plus, comme chacun des quatre segments obtenus est de la même longueur que les autres, les transformations de l'IFS subdivisent l'espace paramétrique en quatre parties égales :

$$\forall t \in \mathbb{R}, \quad \mathcal{T}_0(t) = \frac{t}{4} ; \quad \mathcal{T}_1(t) = \frac{t}{4} + \frac{1}{4} ; \quad \mathcal{T}_2(t) = \frac{t}{4} + \frac{1}{2} ; \quad \mathcal{T}_3(t) = \frac{t}{4} + \frac{3}{4} ;$$

Deux de ces quatre transformations possèdent un point fixe intéressant du point de vue de la génération d'adresses :

$$\mathcal{P}_0(t) = 0 ; \quad \mathcal{P}_3(t) = 1$$

En deux itérations, 16 segments sont générés et donc 17 points correspondant aux différents paramètres  $t = \{0, \frac{1}{16}, \dots, \frac{15}{16}\}$ . Les adresses générées sont au nombre de 32 : 1 seule pour les points  $t = 0$  et  $t = 1$  et deux pour tous les autres points. Ces adresses sont toutes de la forme  $\mathcal{T}_i \mathcal{T}_j \mathcal{P}_k$  avec  $i, j \in \{0 \dots 3\}$  et  $k \in \{0, 3\}$

Comme il y a discontinuité de tangente en chaque point, les adresses-doubles sont conservées pour le calcul des tangentes.

Pour définir l'IFS, trois points de contrôle sont nécessaires : les deux extrémités d'un segment à interpoler et un autre point pour donner l'allure générale de la courbe. L'espace barycentrique est donc  $\mathbf{BI}^3$  et les transformations correspondantes sont représentées par les matrices suivantes :

$$\mathcal{M}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 2/3 & 1/3 & 0 \\ 2/3 & 0 & 1/3 \end{pmatrix} \quad \mathcal{M}_1 = \begin{pmatrix} 2/3 & 0 & 1/3 \\ 2/3 & 1/3 & 0 \\ 1/3 & 1/3 & 1/3 \end{pmatrix} \quad \mathcal{M}_2 = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 2/3 \\ 1/3 & 0 & 2/3 \end{pmatrix} \quad \mathcal{M}_3 = \begin{pmatrix} 1/3 & 0 & 2/3 \\ 0 & 1/3 & 2/3 \\ 0 & 0 & 1 \end{pmatrix}$$

Les points-fixes et les tangentes nécessaires pour la génération de la tessellation de l'attracteur sont :

$$\mathcal{P}_0 = (1; 0; 0) ; \quad \vec{\tau}_0 = (-1; 0; 1) ; \quad \mathcal{P}_1 = (0; 0; 1) ; \quad \vec{\tau}_1 = (1; 0; -1)$$

Les 17 points fixes et les 32 tangentes sont calculés dans l'espace barycentrique en suivant les adresses. Cette liste de combinaisons est ensuite plongée dans les différents polygones de contrôle pour construire une tessellation de l'attracteur ainsi que les tangentes en chaque point. Un schéma représentant la méthode globale est donné en Figure 2.9.

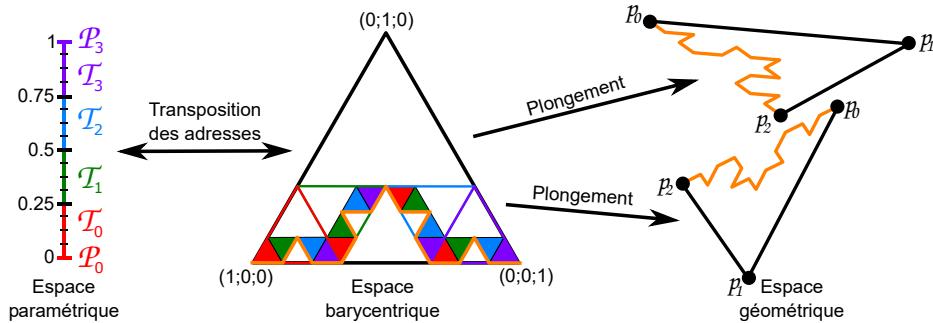


FIGURE 2.9 – Schéma représentant la méthode globale du calcul d'une tessellation de l'attracteur d'un IFS barycentrique pour un ensemble de contrôle donné.

Les principaux avantages de l'utilisation d'IFS barycentriques par rapport aux IFS de [Gol04, SLG05] et aux IFS projetés [ZT96] sont les suivants :

- Séparation stricte des différentes étapes en différents espaces. Ainsi le même processus itératif est utilisé dans l'espace des paramètres pour construire une liste d'adresse et dans l'espace barycentrique pour générer une tessellation de l'attracteur, puis cette tessellation est plongée dans l'espace géométrique.
- La tessellation est calculée une fois pour toutes dans l'espace barycentrique puis plongée de multiples fois dans l'espace géométrique. Ainsi, ajouter ou modifier un ensemble de contrôle ne nécessite pas de nouvelles itérations, seulement le calcul d'une projection supplémentaire.

### 2.3.6 IFS barycentriques de courbes CGAO usuelles

Dans les travaux de Goldman [Gol04] puis de Schaeffer *et al.* [SLG05], la notion d'espace barycentrique n'est pas évoquée. Dans l'article de Zaïr et Tosan [ZT96] il est appelé espace d'itération. Pourtant, tous ces résultats peuvent être transposés dans une approche basée IFS barycentrique quasiment sans modification.

#### Les courbes de Bézier

L'idée de Bézier était : "Au lieu de déformer la courbe, ou une famille de courbes, il vaut mieux faire subir une distortion générale à l'espace dans lequel on les a incluses". Pour cela, il cherche à définir des courbes dont la forme est contrôlée par un ensemble de points de contrôle. Avec l'aide de Claude Riaux, il définit que l'influence des points de contrôle sur la courbe évolue en fonction du paramètre de la courbe selon les polynômes de Bernstein.

Soit une courbe de Bézier  $\mathcal{C}_d(t)$  de degré  $d$  définie par un polygone de contrôle  $\mathbf{P}_0 = \{p_0^0 \dots p_d^0\}$ . Tout point de la courbe est défini par la formule suivante :

$$\mathcal{C}_d(t) = \sum_{i=0}^d \mathbf{B}_i^d(t) p_i^0$$

où  $\mathbf{B}_i^d$  est le polynôme de Bernstein :

$$\mathbf{B}_i^d(t) = \binom{d}{i} t^i (1-t)^{d-i} = \frac{d!}{i!(d-i)!} t^i (1-t)^{d-i}$$

Il est plus courant de définir les points d'une courbe de Bézier par récurrence en utilisant les règles suivantes :

$$\begin{aligned} \mathcal{C}_d(t) &= p_0^d(t) \\ p_i^d(t) &= (1-t)p_i^{(d-1)}(t) + tp_{i+1}^{(d-1)}(t) \end{aligned}$$

Le calcul est souvent représenté par un arbre. Un exemple pour les courbes de Bézier cubique est donné en Figure 2.10.

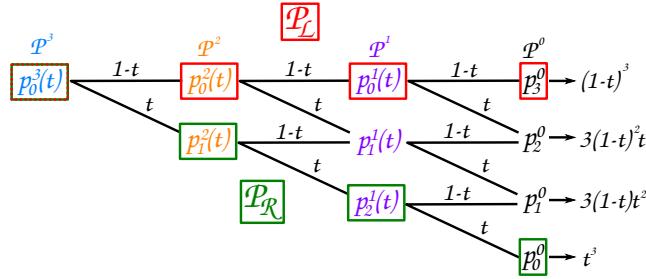


FIGURE 2.10 – Représentation sous forme de *blossoming* du calcul des coefficients d'une courbe de Bézier cubique. Le calcul est celui du point  $\mathcal{C}_3(t) = p_0^3(t)$  (représenté en cyan) à partir du polygone de contrôle (en noir). Les polygones intermédiaires sont en violet et orange. Les points des polygones  $\mathbf{P}_L$  et  $\mathbf{P}_R$  sont respectivement encadrés en rouge et en vert. Les couleurs sont les mêmes que dans la Figure 2.11.

**L'algorithme de Casteljau** utilisé classiquement pour visualiser les courbes de Bézier est basé sur le fait que chaque partition d'une courbe de Bézier est elle-même une courbe de Bézier. Cet algorithme itératif a pour objectif de définir la courbe par l'union de deux courbes qui sont les parties gauche et droite de la courbe, puis de réitérer sur ces deux courbes. Pour cela, le polygone de contrôle  $\mathbf{P}_0$  de la courbe globale est transformé en deux nouveaux polygones  $\mathbf{P}_L$  et  $\mathbf{P}_R$  qui génèrent respectivement les sous-courbes gauche et droite. A partir d'un certain nombre d'itérations, les points des polygones de contrôle sont suffisamment nombreux et proches de la courbe pour afficher ces polygones comme une approximation de la courbe.

Bien que l'algorithme de Casteljau permette de découper les courbes de Bézier en un point quelconque de la courbe, il est d'usage de découper la courbe en son milieu ( $t = 1/2$ ). En fixant  $t = 1/2$  dans la Figure 2.10, il apparaît que chaque point d'un polygone  $\mathbf{P}_0^i$  est le milieu d'une arête de  $\mathbf{P}_0^{(i-1)}$ . En itérant  $\mathbf{P}_0^0$  jusqu'à  $\mathbf{P}_0^d = \{p_0^d\}$  plusieurs polygones intermédiaires dits "de construction" apparaissent. Les polygones de contrôle des courbes gauche  $\mathbf{P}_{\mathcal{L}}$  et droite  $\mathbf{P}_{\mathcal{R}}$  sont respectivement définis par le premier et le dernier point de chacun de ces polygones intermédiaires :

$$\begin{aligned}\mathbf{P}_{\mathcal{L}} &= \{ p_0^i \}, i \in \{0 \dots d\} \\ \mathbf{P}_{\mathcal{R}} &= \{ p_j^{d-j} \}, j \in \{0 \dots d\}\end{aligned}$$

Un exemple de l'algorithme de De Castlejau est donné en Figure 2.11.

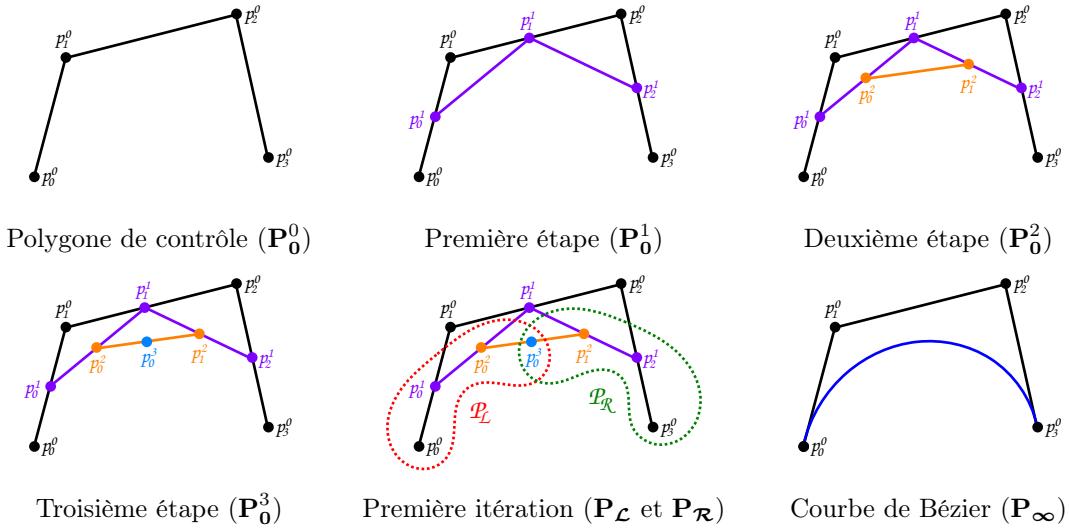


FIGURE 2.11 – Génération d'une courbe de Bézier cubique par l'algorithme de De Casteljau. Le polygone de contrôle est en noir, les polygones intermédiaires en violet, orange, et cyan. Les deux sous-polygones générés après la première itération sont en rouge pour  $\mathbf{P}_{\mathcal{L}}$  et en vert pour  $\mathbf{P}_{\mathcal{R}}$  et la courbe de Bézier limite  $\mathbf{P}_{\infty}$  est en bleu. Les coefficients pour calculer les différents points sont trouvables dans la Figure 2.10.

Les itérations de l'algorithme de De Casteljau sont classiquement calculées par des matrices de transformations  $\mathcal{M}_{\mathcal{L}}$  et  $\mathcal{M}_{\mathcal{R}}$  qui associe au polygone de contrôle  $\mathbf{P}$  d'une courbe de Bézier, les deux sous-polygones de contrôle  $\mathbf{P}_{\mathcal{L}}$  et  $\mathbf{P}_{\mathcal{R}}$ . Par exemple, pour les courbes de Bézier cubiques, les matrices sont les suivantes :

$$\mathcal{M}_{\mathcal{L}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1/4 & 0 \\ 1/8 & 3/8 & 3/8 & 1/8 \end{pmatrix}; \quad \mathcal{M}_{\mathcal{R}} = \begin{pmatrix} 1/8 & 3/8 & 3/8 & 1/8 \\ 0 & 1/4 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Le fait que la dernière ligne de  $\mathcal{M}_{\mathcal{L}}$  et la première de  $\mathcal{M}_{\mathcal{R}}$  soient identiques assure le raccord entre les deux sous-courbes. De plus, comme les courbes de Bézier interpolent les points extrêmes du polygone, il est naturel que les points fixes de  $\mathcal{M}_{\mathcal{L}}$  et  $\mathcal{M}_{\mathcal{R}}$  soient respectivement  $p_0^0$  et  $p_d^0$ . Remarquez également que la matrice  $\mathcal{M}_{\mathcal{L}}$  est triangulaire inférieure et que le triangle est celui de Pascal ; et que  $\mathcal{M}_{\mathcal{R}}$  est obtenue par "rotation" de  $\mathcal{M}_{\mathcal{L}}$ .

A partir de cette représentation et du fait que chaque sous-partie d'une courbe de Bézier est elle-même une courbe de Bézier (il y a auto-similarité), il apparaît que l'algorithme de De Casteljau est un IFS comme précisé dans [Gol04, ZT96]. Dans le cas présent, l'IFS barycentrique de Bézier est défini par les deux transformations  $\mathcal{L}$  et  $\mathcal{R}$  à la fois dans l'espace des paramètres comme le découpage diadique (en 0.5 de  $[0; 1]$ ) et dans l'espace barycentrique comme l'application des matrices de Casteljau  $\mathcal{M}_{\mathcal{L}}$  et  $\mathcal{M}_{\mathcal{R}}$ . Un schéma global de l'IFS barycentrique est donné en Figure 2.12.

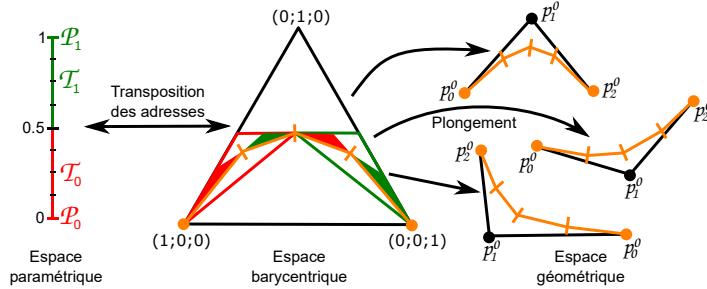


FIGURE 2.12 – IFS barycentrique d'une courbe de Bézier quadratique. Les transformations  $T_0$  et  $T_1$  sont représentées en rouge et vert. L'espace paramétrique est subdivisé de façon diadique. Dans l'espace barycentrique, la première itération est représentée par des triangles vides et la seconde par des triangles pleins.

### IFS de B-Splines uniformes

Les B-splines uniformes [Sch46] sont des courbes définies par morceaux, tout comme peuvent le faire des splines d'Hermite ou de Catmull-Rom. Une B-Spline uniforme de degré  $d$ , composée de  $m$  morceaux est une courbe unique définie par un polygone de contrôle composé de  $(d+m)$  points de contrôle où chaque ensemble de  $(d+1)$  points consécutifs du polygone définit un morceau de la courbe. Comme chaque morceau est traité indépendamment des autres, toutes les B-Splines présentées dans ce manuscrit sont composées d'un seul morceau et contrôlées par un polygone de  $(d+1)$  points. Pour le moment, seules les B-Splines uniformes sont expliquées, leur généralisation que sont les B-Splines (Rationnelles) Non-Uniformes (NU(R)BS) font l'objet du [Chapitre 4](#). La formule récursive pour calculer une B-Spline uniforme de degré  $d$  est la suivante :

$$\begin{aligned} C_d(t) &= \sum_{i=0}^{d+1} (N_i^d p_i) \\ N_j^{d>0}(t) &= \frac{(t-j)N_j^{(d-1)}(t) + (j+d+1-t)N_{(j+1)}^{(d-1)}(t)}{d} \\ N_j^0(t) &= 1 \text{ si } t \in [j ; (j+1)] \text{ et } 0 \text{ sinon} \end{aligned}$$

La récursivité de la formule est généralement représentée sous la forme de l'arbre en [Figure 2.13](#). Comme un seul  $N_j^0$  est actif à la fois (tous les autres sont nuls), seules les fonctions d'indice et exposant  $N_i^d, i \in \{j-d, \dots, j\}$  sont actives. Dans une variante optimisée, les branches de l'arbre de récurrence correspondant à des fonctions inactives sont élaguées.

Tout comme l'algorithme de De Casteljau avec les courbes de Bézier, l'**algorithme de Chaikin** [Cha74] décrit les B-Splines quadratiques uniformes d'une manière itérative appelée *corner-cutting* (notion étudiée plus tôt par Georges de Rahm pour définir la forme arrondie d'un manche de marteau [DR47]). Comme chaque restriction d'une B-Spline est elle-même une B-Spline, l'idée est de trouver, à partir d'un polygone de contrôle  $\mathbf{P}_0$ , un autre polygone ayant un point de plus  $\mathbf{P}_1$  qui définit la même courbe mais cette fois en deux morceaux. En répétant l'algorithme itérativement sur le nouveau polygone permet, après une infinité d'itérations, que tous les points du polygone appartiennent à la B-Spline quadratique. Chaque itération est décrite par Chaikin [Cha74] de la manière suivante (dont les coefficients sont généralement dit "de Chaikin") :

$$\begin{aligned} p'_{2i} &= \frac{3}{4}p_i + \frac{1}{4}p_{i+1} \\ p'_{2i+1} &= \frac{1}{4}p_i + \frac{3}{4}p_{i+1} \end{aligned}$$

Ces coefficients peuvent être retrouvés dans l'arbre de la [Figure 4.1](#) : il s'agit des coefficients des branches entre les  $N_i^2$  et les  $N_j^1$  lorsque  $t = 2.5$ . Ce principe est appelé *mid-knot insertion* (insertion au milieu entre chaque noeud) et est détaillé dans le [Chapitre 4](#). Un exemple de la construction d'une B-Spline quadratique uniforme par l'algorithme de Chaikin est donné en [Figure 2.14](#)

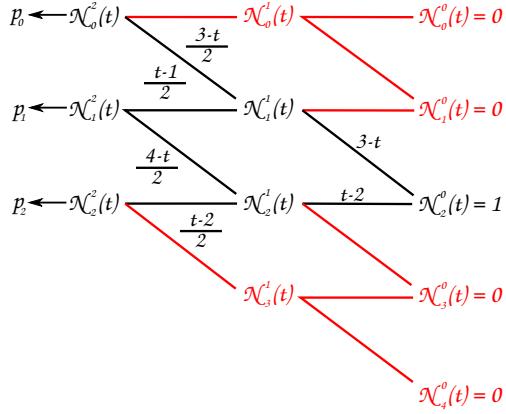


FIGURE 2.13 – Arbre de la construction par récurrence des fonctions de base B-Splines uniformes. Les feuilles en rouge représentent les fonctions nulles. Toutes les branches qui conduisent à une fonction nulle n’ont pas de coefficient car celui-ci est annulé par la fonction à laquelle la branche conduit. Est également mis en rouge tout noeud dont toutes les branches qui partent sont rouges car la fonction à laquelle il correspond est nulle. L’arbre noir qui apparaît à l’inverse de l’arbre global est celui du *blossoming* qui est présenté dans le [Chapitre 4](#).

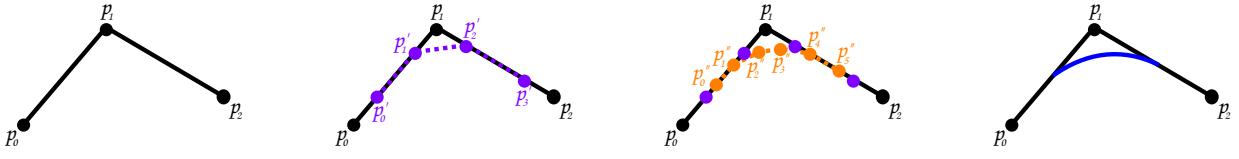


FIGURE 2.14 – Exemple de l’application de l’algorithme de Chaikin. Le polygone de contrôle  $\mathbf{P}_0 = [p_0 \dots p_2]$  est représenté en noir, les polgones de contrôle obtenus après une itération  $\mathbf{P}_1 = [p'_0 \dots p'_3]$  et deux itérations  $\mathbf{P}_0 = [p''_0 \dots p''_5]$  sont en rouge et vert. La courbe limite  $\mathbf{P}_\infty$  est en bleu.

Plutôt que de définir une B-Spline quadratique composée de deux morceaux, il est courant de définir deux B-Splines d’un seul morceau et d’appliquer l’algorithme de manière récursive sur les deux nouveaux polygones. Ainsi le nouveau polygone de contrôle  $\mathbf{P}' = [p'_0 ; p'_1 ; p'_2 ; p'_3]$  est décomposé en deux polygones de contrôle : gauche  $\mathbf{P}_{\mathcal{L}} = [p'_0 ; p'_1 ; p'_2]$  et droite  $\mathbf{P}_{\mathcal{R}} = [p'_1 ; p'_2 ; p'_3]$ . Ceci implique que, tout comme les courbes de Bézier, les B-Splines quadratiques peuvent être définies par un IFS composé de deux transformations :  $\mathcal{L}$  et  $\mathcal{R}$  [ZT96, SLG05].

**L’algorithme de Lane-Riesenfeld** [LR80] est une généralisation de l’algorithme de Chaikin. Cet algorithme démontre que toute B-Spline uniforme peut être construite de manière itérative où chaque itération est composée d’une étape de dédoublement et d’un certain nombre d’étapes de moyennage. L’étape de dédoublement consiste simplement à dédoubler chaque point du polygone de contrôle, puis à relier chaque sommet à son clone et chaque clone au sommet original suivant. L’étape de moyennage consiste en l’insertion d’un sommet au milieu de chaque arête, ces sommets sont ensuite reliés en respectant l’ordre des arêtes. Les deux opérateurs de dédoublement  $\mathbf{D}$  et de moyennage  $\mathbf{M}$  sont définis telles que :

$$\begin{aligned}\mathbf{D} &: [p_0, p_1 \dots p_n] \mapsto [p_0, p_0, p_1, p_1, \dots, p_n, p_n] \\ \mathbf{M} &: [p_0, p_1 \dots p_n] \mapsto \left[ \frac{p_0 + p_1}{2}, \dots, \frac{p_{n-1} + p_n}{2} \right]\end{aligned}$$

Chaque itération de l’algorithme de construction d’une B-Spline uniforme de degré  $d$  est de la forme (un exemple d’une itération dans le cas cubique est donné en [Figure 2.15](#)) :

$$\mathbf{P}_{i+1} = \mathbf{M}^d \circ \mathbf{D} \mathbf{P}_i$$

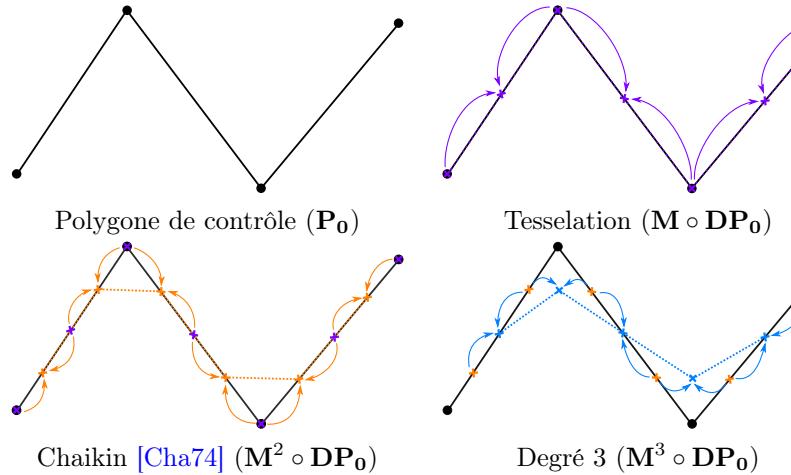


FIGURE 2.15 – Génération des B-Splines uniformes par l'algorithme de Lane-Riesenfeld

Et donc la courbe limite est définie depuis le polygone de contrôle :

$$\mathbf{P}_\infty = (\mathbf{M}^d \circ \mathbf{D})^\infty \mathbf{P}_0$$

L'opérateur de dédoublement  $\mathbf{D}$ , comme son nom l'indique, double le nombre de points du polygone. Chaque application de l'opérateur de moyennage  $\mathbf{M}$  retire un point de ce polygone (parce qu'il y a un point par arête). En sachant que le polygone de départ  $\mathbf{P}_0$  est de taille  $(d+1)$ , le polygone de l'itération suivante  $\mathbf{P}_1$  est de taille  $(d+2)$  ce qui implique une B-Spline uniformes composée de deux morceaux. Cette B-Spline uniformes peut donc être découpé en deux B-Splines uniformes d'un seul morceau définies par les polygones de contrôle gauche  $\mathbf{P}_{\mathcal{L}} = [p'_0 \dots p'_d]$  et droite  $\mathbf{P}_{\mathcal{R}} = [p'_1 \dots p'_{d+1}]$ . Le résultat de [SLG05] peut donc être généralisé aux B-Splines uniformes de degré quelconque : toute B-Spline uniforme peut être décrite par un IFS barycentrique dont les transformations  $\mathcal{L}$  et  $\mathcal{R}$  correspondent à la subdivision diadique dans l'espace des paramètres et aux matrices  $\mathcal{M}_{\mathcal{L}}$  et  $\mathcal{M}_{\mathcal{R}}$  dans l'espace barycentrique. Ce résultat a été obtenu par Zaïr [ZT96], dans le cadre plus général des courbes définies dans une base polynomiale.

La méthode utilisée pour générer les matrices de transformation  $\mathcal{M}_{\mathcal{L}}$  et  $\mathcal{M}_{\mathcal{R}}$  de l'IFS des B-Splines de degré  $d$  est directement déduite de l'algorithme de Lane-Riesenfeld :

- La matrice  $\mathcal{M}_{\mathbf{D}}(d+1)$  qui correspond à l'opérateur de dédoublement  $\mathbf{D}$  d'un polygone de  $(d+1)$  points est construite. Cette matrice, uniquement composée de 0 et de 1, est de taille  $(2d+2) \times (d+1)$ .
- Les matrices  $\mathcal{M}_{\mathbf{M}}(i)$ ,  $i \in \{(d+2) \dots (2d+1)\}$  qui correspondent à l'opérateur de moyennage  $\mathbf{M}$  d'un polygone de taille  $(i+1)$  vers un polygone de taille  $i$  sont également construites. Ces matrices de taille  $i \times (i+1)$  ont pour coefficients uniquement des 0 et des  $1/2$ .
- La matrice globale est calculée par la formule suivante :  $\mathcal{M}_{\mathbf{M}}(d+2) \dots \mathcal{M}_{\mathbf{M}}(2d+1) \mathcal{M}_{\mathbf{D}}(d+1)$ . Cette matrice est de taille  $(d+2) \times (d+1)$  et correspond à la transformation qui associe un polygone  $\mathbf{P}$  à celui de l'itération suivante  $\mathbf{P}'$ .
- Cette matrice globale est découpée en deux sous-matrices  $\mathcal{M}_{\mathcal{L}}$  et  $\mathcal{M}_{\mathcal{R}}$  qui sont respectivement les  $(d+1)$  premières et les  $(d+1)$  dernières lignes.

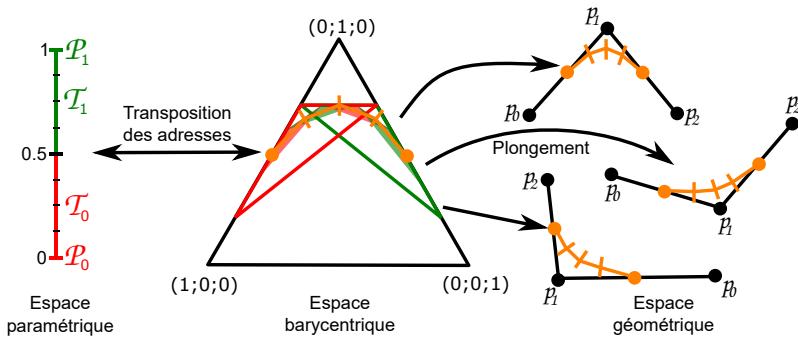
Les matrices  $\mathcal{M}_{\mathbf{D}}$  et  $\mathcal{M}_{\mathbf{M}}$  sont de la forme suivante :

$$\mathcal{M}_{\mathbf{D}} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}; \quad \mathcal{M}_{\mathbf{M}} = \begin{pmatrix} 1/2 & 1/2 & 0 & \dots & \dots & 0 \\ 0 & 1/2 & 1/2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1/2 & 1/2 \end{pmatrix}$$

Par exemple, les matrices de l'IFS de Chaikin sont :

$$\begin{aligned} \mathcal{M}_M(4)\mathcal{M}_M(5)\mathcal{M}_D(3) &= \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \end{pmatrix} \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 1/2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 3/4 & 1/4 & 0 \\ 1/4 & 3/4 & 0 \\ 0 & 3/4 & 1/4 \\ 0 & 1/4 & 3/4 \end{pmatrix} \\ \Rightarrow \mathcal{M}_L &= \begin{pmatrix} 3/4 & 1/4 & 0 \\ 1/4 & 3/4 & 0 \\ 0 & 3/4 & 1/4 \end{pmatrix}; \quad \mathcal{M}_R = \begin{pmatrix} 1/4 & 3/4 & 0 \\ 0 & 3/4 & 1/4 \\ 0 & 1/4 & 3/4 \end{pmatrix} \end{aligned}$$

Un schéma de l'IFS barycentrique de Chaikin est donnée en [Figure 2.16](#)



**FIGURE 2.16 – IFS barycentrique d'une B-Spline quadratique dit "IFS barycentrique de Chaikin".** Les transformations  $T_0$  et  $T_1$  sont représentées en rouge et vert. L'espace paramétrique est subdivisé de façon diadique. Dans l'espace barycentrique, la première itération est représentée par des triangles vides et la seconde par des triangles pleins et transparents.

## 2.4 Automates d'IFS contrôlés (CIFS)

Les IFS contrôlés (CIFS) sont une extension des IFS qui permet de générer de nouvelles formes d'attracteur. Jusqu'à présent, les transformations des IFS étaient construites de manière à transformer un objet en un objet de même nature (*e.g.* les transformations de Sierpiński associent un triangle à un triangle, celle de Lane-Riesenfeld un polygone de contrôle à un autre). Cette règle n'est plus valable dans le cadre des CIFS où la nature de l'objet peut être modifiée au cours de la transformation, qui cependant reste contractante. Par exemple, les transformations de la [Figure 2.17](#) modifient la nature des objets "triangle" et "carré". Les CIFS permettent de représenter ce genre de transformations sous la forme d'un automate fini déterministe.

### 2.4.1 Rappel sur les automates finis déterministes

Avant de définir les automates de CIFS, il faut rappeler les règles qui régissent les automates finis déterministes dont les automates CIFS sont un sous-ensemble. Un exemple d'automate est présent en [Figure 2.18](#). Ces automates sont composés :

- d'un **ensemble fini d'états**  $Q = \{q_0 \dots q_{k-1}\}$  dont en particulier l'**état de départ**  $q_0$
- $F \subset Q$  l'**ensemble des états finaux**
- d'un **alphabet**  $\Sigma = \{\alpha_0 \dots \alpha_{n-1}\}$  qui définit un ensemble de **mots**, appelé **langage de l'alphabet** qui sont toutes les combinaisons possibles des **lettres**  $\alpha_i$
- d'un **ensemble de transition**  $\delta : Q \times \Sigma \mapsto Q$  qui permettent le passage d'un état à l'autre

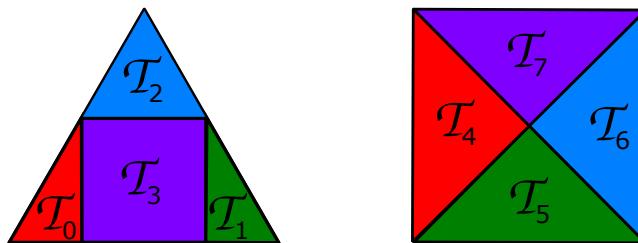


FIGURE 2.17 – Exemples de transformations contractantes modifiant la nature de l'objet. Les transformations  $\mathcal{T}_0$ ,  $\mathcal{T}_1$ , et  $\mathcal{T}_2$  associent un triangle à un autre,  $\mathcal{T}_4$  à un carré, et les transformations  $\mathcal{T}_5$ ,  $\mathcal{T}_6$ ,  $\mathcal{T}_7$ , et  $\mathcal{T}_8$  un carré à un triangle.

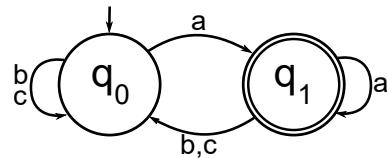


FIGURE 2.18 – Exemple d'un automate fini déterministe. Les états sont représentés par des ronds et étiquetés  $q_0$  et  $q_1$ . L'alphabet est composé des lettres a, b, et c. Les transitions sont représentées par des flèches, étiquetées par les lettres des transitions, qui relient l'état de départ et l'état d'arrivée de la transition. L'état de départ de l'automate  $q_0$  est désigné par une flèche entrante et l'état final  $q_1$  par un double-rond.

Un mot est considéré comme **reconnu** (ou accepté) si, en partant de l'état de départ et en changeant d'état en suivant les transitions indiquées par la liste des lettres du mot, les transitions existent toujours et le dernier état atteint est un des états finaux. L'ensemble des mots reconnus par l'automate est appelé **langage de l'automate**. Par exemple, le langage de l'automate de la Figure 2.18 est l'ensemble des mots finis d'au moins une lettre, composé uniquement des lettres a, b, et c, et qui se terminent par un a.

## 2.4.2 Construction des automates CIFS

La construction de l'automate se fait en deux étapes : d'abord la structure de l'automate est définie à partir de l'ensemble des transformations du CIFS puis l'état final et les transitions qui conduisent à lui sont ajoutés.

### Structure générale de l'automate

Pour les automates CIFS :

- les **états** sont l'ensemble des **objets** de natures différentes (et plus précisément des attracteurs) du CIFS,
- l'**alphabet** de l'automate est le même que celui qui génère les **adresses**
- les **transitions** sont l'ensemble des **transformations** du CIFS,
- l'**état de départ** est celui qui contraint la **forme globale de l'attracteur**.

Lors de la construction d'un automate CIFS, un état est créé pour chaque nature d'objet présente dans les transformations. Ensuite, pour chaque transformation, une transition est ajoutée à l'automate. Cette transition a pour état de départ et état d'arrivée les deux types d'objet impliqués dans la transformation (potentiellement deux fois le même type) et elle a pour lettre associée le nom de la transformation. Un exemple est donné en Figure 2.19

### L'état final "point de l'attracteur" : les points fixes du CIFS

Classiquement, les automates de CIFS sont utilisés pour définir les compositions autorisées de transformations et ainsi contrôler plus finement le processus de subdivision de l'attracteur. Les mots acceptés par l'automate sont des mots infinis (l'ensemble des adresses de tous les points de l'attracteur). Ici, les adresses sont volontairement limitées à des mots finis se terminant  $\mathcal{P}_i$ , car ils sont effectivement calculables. La tesselation sera construite à partir de cet ensemble de points qui est un sous ensemble dense de points de l'attracteur. Une fois la structure globale de l'automate construite, l'état final qui correspond aux points de l'attracteur et les transitions correspondantes

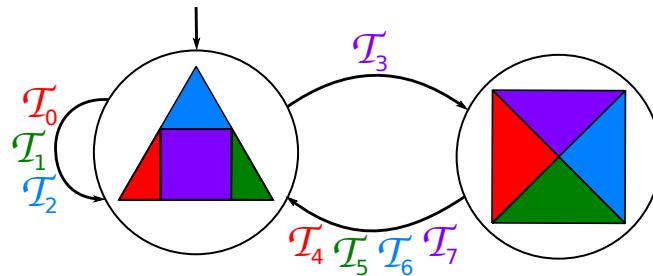


FIGURE 2.19 – Exemple d'un automate CIFS incomplet composé de deux états et de huit transformations  $T_i$ .

sont ajoutés. Pour commencer n'importe quel IFS peut être représenté sous la forme d'un CIFS à un seul état. Réciproquement, dans un automate CIFS, tout sous-automate composé d'un unique état et des transitions qui bouclent sur cet état est un IFS. Les états qui ont des transformations qui bouclent sont dits **stationnaires**.

Dans les sections précédentes, il est expliqué comment calculer le point-fixe  $P_i$  d'une transformation  $T_i$  d'un IFS. Cette méthode est appliquée sur toutes les transformations de l'automate qui sont des transformations d'un IFS (donc toutes celles qui bouclent sur un état). L'ensemble des transformations points-fixes ainsi calculées sont rajoutées à l'automate en tant que transition vers un nouvel état "point-fixe" ou "point de l'attracteur" qui est le seul état final de l'automate (voir Figure 2.20). Le langage reconnu par l'automate correspond à l'ensemble des adresses valides.

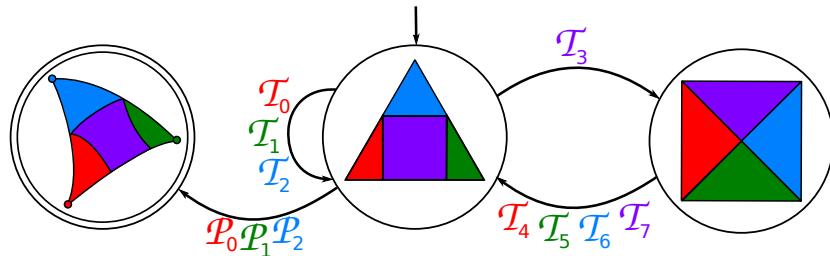


FIGURE 2.20 – Exemple d'automate CIFS complet.

Dans le cas d'une composition d'une transformation qui associe un objet de nature **A** à un objet de nature **B** avec une transformation qui associe **B** à **A**, la transformation globale ne change pas la nature de l'objet. Comme les deux transformations sont contractantes, leur composition l'est aussi. Ainsi la transformation composée peut également être considérée comme une transformation contractante qui conserve la nature de l'objet : son point-fixe et ses dérivées peuvent être étudiés. Dans la pratique, les compositions de type "aller-retour" ou plus généralement les compositions circulaires de transformations ne présentent que rarement d'intérêt mais ce sont également des transformations stationnaires de l'IFS.

### 2.4.3 Opérations sur les automates

Les automates CIFS possèdent un certain nombre d'opérations qui permettent, entre autres, de composer des automates pour composer les attracteurs des CIFS correspondants.

#### Produit tensoriel d'automate

Une méthode usuelle de description de surfaces est par produit-tensoriel de deux courbes. Ce produit-tensoriel est défini pour les IFS et il convient de le définir pour les automates CIFS. La notion de "produit-tensoriel" d'automates finis déterministes n'a pas vraiment de sens en dehors du cas où ils représentent des objets géométriques mais elle reste tout de même applicable.

Soient **A** et **B** deux automates finis déterministes dont les états sont respectivement  $\{a_0 \dots a_{m-1}\}$  et  $\{b_0 \dots b_{n-1}\}$  et dont les transitions sont  $\{\alpha_0 \dots \alpha_{p-1}\}$  et  $\{\beta_0 \dots \beta_{q-1}\}$ . Soit **C** l'automate "produit-tensoriel" de **A** et **B**. L'état initial de **C** est la composition des états initiaux de **A** ( $a_0$ ) et **B** ( $b_0$ ) et est étiqueté  $(a_0; b_0)$ . Pour chaque paire de transitions  $\alpha_i$  (qui associe  $a_0$  à  $a_k$ ) et  $\beta_j$  ( $b_0$  à  $b_\ell$ ), une nouvelle transition de **C**,  $\gamma_{(i,j)}$  est créée entre l'état initial  $(a_0; b_0)$  vers un potentiel nouvel état  $(a_k; b_\ell)$ . Si cet état existe déjà, seule la transition est ajoutée à l'automate ; sinon le nouvel état est également ajouté et ses transitions sont étudiées avec la même méthode que pour l'état initial. Lorsque tous les états de **C** ont été étudiés, la construction de l'automate est terminée. Tous les états issus de deux états finaux de **A** et de **B** sont les états finaux de **C**. Un exemple est donné en Figure 2.21

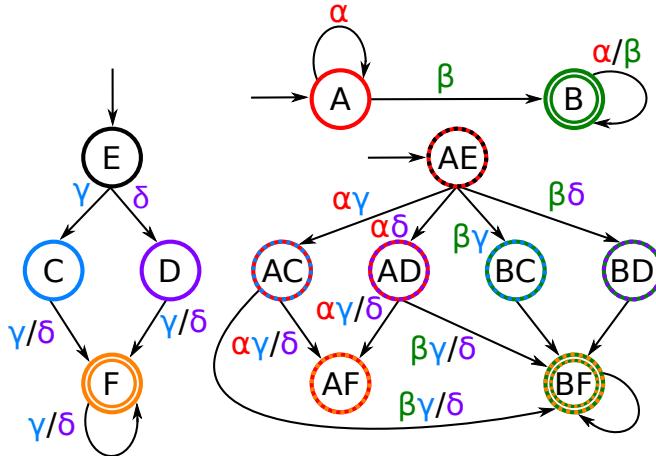


FIGURE 2.21 – Exemple de produit-tensoriel d'automates finis déterministes. Le premier automate (en haut) est composé des états A (départ) et B (final) et des lettres  $\alpha$  et  $\beta$  et le second (à gauche) des états C, D, E (départ), et F (final). L'état de départ du produit tensoriel est donc AE. Les états issus de AE sont AC, AD, BC, et BD. En répétant l'étude des états, l'ensemble de l'automate est généré jusqu'à l'état final BF. Les lettres séparées par un / indiquées sur les transitions sont l'une ou l'autre. Les transitions sans étiquette sont composées de n'importe quel couple de lettres.

A partir de cette méthode de construction, les propriétés suivantes apparaissent :

- **C** est aussi un automate fini déterministe dont le nombre d'états et de transitions ont une borne supérieure égale à  $(m.n)$  et  $(p.q)$
- Si **A** et **B** sont des automates complets (*i.e.* si depuis chaque état, il existe une transition pour chaque lettre de l'alphabet), **C** est également complet
- La composition de deux états stationnaires est un état stationnaire
- La composition de deux états finaux est un état final.

### Raccord entre automates de courbes et de surfaces

En CGAO, il est souvent nécessaire de raccorder des courbes ou des surfaces sans modifier les objets de départ. Pour cela, un **raccord** est ajouté entre ces deux objets. La méthode la plus simple pour raccorder deux courbes A et B est de relier deux points extrêmes par un segment de droite R (voir Figure 2.22).

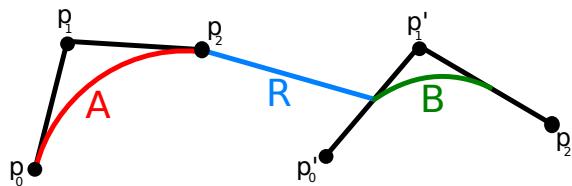


FIGURE 2.22 – Exemple de raccord  $\mathcal{C}_0$ -continu au niveau de ses jonctions

Cette méthode est loin d'être satisfaisante : les **jonctions** entre le raccord et les courbes ne respectent pas les propriétés de ces courbes. Pour cela, il faut que le raccord conserve les tangentes et les courbures au niveau des **points de jonction**.

Podkorytov *et al.* [PGSL13] proposent la solution suivante pour raccorder deux courbes représentées par des automates CIFS **A** et **B** : un automate de raccord **R** est ajouté entre ces deux automates (voir Figure 2.23). Le principe est de construire le raccord en prolongeant chacune des deux courbes tout en les transformant progressivement l'une dans l'autre. Le raccord possède alors de chaque côté les propriétés de la courbe qu'il prolonge et assure ainsi une jonction "propre" entre les deux courbes.

Cet automate possède trois transformations :

- "raccord gauche" ( $\mathbf{R}_L$ ) qui transforme le raccord en une courbe de type **A** pour assurer la jonction avec la courbe de l'automate **A**
- "raccord droit" ( $\mathbf{R}_R$ ) qui transforme le raccord en une courbe de type **B** pour assurer la jonction avec la courbe de l'automate **B**
- "milieu" ( $\mathbf{R}_M$ ) qui transforme le raccord en une sous-partie de lui-même pour assurer la continuité interne du raccord.

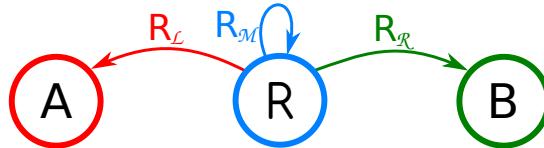


FIGURE 2.23 – Structure globale d'un automate de raccord. Les trois transformations de l'automate **R** conduisent aux états des automates **A** et **B** à raccorder et à lui-même.

La méthode se généralise aux surfaces.

### En résumé :

- Les Systèmes de Fonctions Itérés (IFS) sont un outil représentant les fractales comme des ensembles de transformations contractantes.
- Les IFS barycentriques permettent la construction de certaines courbes et des surfaces CGAO classiques (*e.g.* Bézier, B-Splines uniformes) en plongeant l'attracteur de l'IFS depuis l'espace barycentrique vers l'ensemble des points de contrôle.
- Les automates d'IFS contrôlés (CIFS) sont une extension des IFS permettant de construire des objets auto-similaires en composant des objets de différentes natures. Les règles de compositions sont contrôlées par un automate.
- Les automates CIFS sont dotés d'une palette d'outils habituels en CGAO (*e.g.* création de surfaces par produit-tensoriel de courbes, construction de raccord entre deux objets) qui peuvent être utilisés sur un objet de n'importe quelle nature à condition qu'il soit représenté par un automate CIFS.

La construction des automates CIFS des surfaces de subdivision et des surfaces tensorielles NURBS sont présentés dans les prochains chapitres. Grâce à ces nouveaux automates, ces objets usuels de la CGAO peuvent être utilisés dans le formalisme des CIFS pour réaliser des opérations de CGAO faisant intervenir des objets de natures différentes.

# Chapitre 3

## Schémas de subdivision

Ce chapitre explique la conception de l'automate CIFS correspondant à un schéma de subdivision uniforme donné. Ces automates permettent un calcul direct d'une tessellation de la surface limite sans passer par les habituelles applications successives des règles de subdivision des différents schémas.

Pour commencer, un rapide historique de l'évolution des surfaces de subdivision est exposé. Ensuite le schéma de subdivision de Catmull-Clark [CC78], qui est le plus courant, est décrit par ses règles de subdivision, puis par la méthode de génération de la surface limite de Stam [Sta98a], et enfin sous la forme d'un automate CIFS. De cet exemple est déduit un protocole permettant de trouver l'automate CIFS correspondant à un schéma décrit sous forme de règles de subdivision. Ce protocole n'est appliqué que sur les schémas les plus courants mais il semble être applicable à n'importe quel schéma uniforme à condition que la zone d'influence de chaque sommet soit suffisamment compacte. Un exemple de schéma à support non-compact est également donné pour montrer la difficulté combinatoire inhérente à ce genre de schéma.

Les résultats décrits ont été présentés par deux fois à la communauté française : lors des Journées du Groupe de Travail en Modélisation Géométrique (GTMG) [MNLG17a] et des Journées Françaises d'Informatique Graphique (J-FIG) [MNLG17b] ; puis publiés dans le journal de la conférence World Society for Computer Graphics (WSCG, anciennement Winter School of Computer Graphics) [MNLG18b].

### 3.1 Historique des schémas de subdivision

#### 3.1.1 Les schémas originels : Catmull-Clark et Doo-Sabin

Les schémas de subdivision ont été introduits simultanément en 1978 par deux binômes : Catmull et Clark [CC78] et Doo et Sabin [DS78]. Sous leur première forme, les schémas de subdivision sont une extension des surfaces tensorielles B-Splines uniformes aux maillages de topologie arbitraire. En effet, les surfaces B-Splines classiques sont construites par produit tensoriel de deux B-Splines, ce qui implique que le maillage de départ est toujours de **type grille** (dit régulier). A l'inverse, une approche itérative de type *corner-cutting* permet de s'extraire de cette obligation de maillage de type grille grâce à des règles de subdivision qui s'adaptent à la topologie locale du maillage.

Ainsi, l'**opérateur de subdivision S** qui transforme un maillage en un maillage plus fin (*i.e.* plus de sommets et de faces) est défini. Cet opérateur est appliqué de manière itérative sur un maillage de départ  $\mathbf{P}_0$  (appelé **maillage de contrôle**) et les maillages obtenus après un nombre  $i$  d'itérations sont notés  $\mathbf{P}_i$ . L'opérateur de subdivision est contractant ce qui implique qu'après une infinité d'itération, le maillage converge vers une surface  $\mathbf{P}_\infty$  appelée **surface limite** ou **surface de subdivision**. Les propriétés de l'opérateur de subdivision sont résumées par les formules suivantes :

$$\begin{cases} \mathbf{P}_{i+1} &= \mathbf{S}\mathbf{P}_i \\ \mathbf{P}_\infty &= \mathbf{S}^\infty\mathbf{P}_0 \end{cases}$$

La surface limite théorique est composée d'une infinité de faces de taille nulle, ce qui est impossible à représenter par ordinateur. Dans la pratique, la surface limite est représentée par une **tessellation** (*i.e.* par un maillage dont le nombre de faces est fini et dont l'ensemble des sommets appartiennent à la surface limite). Un exemple des étapes successives de subdivision sur un maillage non-régulier est donné en [Figure 3.1](#).

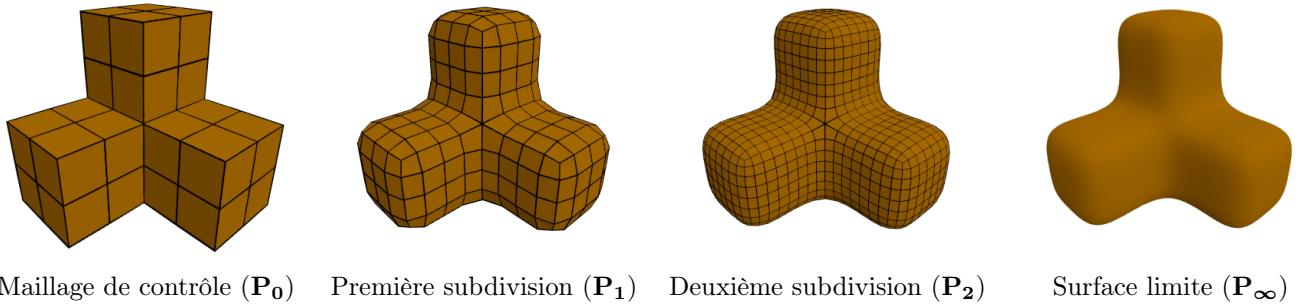


FIGURE 3.1 – Exemple de l’application du schéma de Catmull-Clark sur le maillage "Tetris"

### 3.1.2 Quelques nouveaux schémas

Il faut attendre 1987 pour voir apparaître de nouveaux schémas de subdivision qui comblent les manques laissés par les schémas de Catmull-Clark et Doo-Sabin. Ces schémas conservent l’approche par règles de subdivision des deux premiers schémas mais ces règles sont modifiées en fonction des contraintes du maillage de départ et/ou de l’objectif de la subdivision. Par exemple, Loop déduit de la formulation des Box-Splines un schéma de subdivision pour les maillages triangulaires [Loo87]. Comme l’interpolation est une contrainte classique de la CGAO, la question de schémas de subdivision interpolants se posent également : le schéma Butterfly est le premier à apparaître pour les maillages triangulaires [DLG90] puis celui de Kobbelt pour les maillages quadrangulaires [Kob96]. En une vingtaine d’années, le domaine des surfaces de subdivision peine à s’imposer en CGAO et semble s’essouffler. Mais c’est sans compter sur l’arrivée d’un court-métrage qui dévoile les possibilités qu’elles offrent dans le domaine des films d’animation.

### 3.1.3 Pixar et la démocratisation des schémas de subdivision dans l’animation 3D

En 1986, Steve Jobs rachète *Graphics Group* à *LucasFilm*, renomme le studio *Pixar Animation Studios* et recrute comme directeur technique Edwin Catmull qui travaillera pendant plus de 30 années sur le moteur de rendu : *RenderMan*. Fort du succès de *Toy Story* en 1995, et au cours de la production de *1001 pattes* (1998), Edwin Catmull décide que le studio doit recommencer la conception de court-métrage dont le dernier date de 1989.

L’objectif est de pousser le studio plus loin technologiquement avec le développement d’un personnage principal humain à la fois réaliste en terme de textures de peau et d’animation de vêtements (pour le citer : “*taking human and cloth animation to new heights*”). Pour cela Tony DeRose, Michael Kass, et Tien Truong [DKT98] proposent l’utilisation des surfaces de subdivision de Catmull-Clark qui permettent une animation beaucoup plus expressive que les NURBS utilisées pour *Toy Story*.

En 1997, sort alors *Le Joueur d’échecs* qui gagnera l’Oscar du meilleur court-métrage d’animation l’année suivante grâce à ses prouesses technologiques. C’est à ce moment que les surfaces de subdivision (et en particulier le schéma de Catmull-Clark) s’imposent dans la production de films d’animation et reste encore à ce jour la méthode la plus courante pour produire des modèles aux courbes harmonieuses en animation. Des exemples de résultats sont présentés en Figure 3.2

### 3.1.4 Regain d’intérêt des surfaces subdivision

L’utilisation du schéma de Catmull-Clark pour les films d’animation relance l’intérêt pour le domaine et en quelques années celui-ci progresse bien plus qu’au cours des deux précédentes décennies. Peters et Reif proposent le schéma *Simplest scheme* [PR97] (le plus simple) qui sera, de part ses règles de subdivision, appelé par la communauté le schéma *Mid-edge* (qui relie les milieux des arêtes). Ensuite, Kobbelt propose un schéma triangulaire qui fait une subdivision triadique (*i.e.* chaque arête devient trois arêtes et donc chaque triangle neuf triangles) en deux itérations qu’il nomme  $\sqrt{3}$  [Kob00]. Les maillages qui ne sont ni triangulaires, ni quadrangulaires, sont également pris en compte avec l’arrivée du schéma *Honeycomb* (ruche à miel) pour les maillages hexagonaux [AS02] et du schéma *Quad-Triangle* qui fusionne les schémas de Catmull-Clark et de Loop pour les maillages hybrides (à la fois triangulaires et quadrangulaires) [SL03]. Ces schémas sont détaillés dans les sections suivantes.



Toy Story (1995)

Le Joueur d'échecs (1997)

Coco (2017)

FIGURE 3.2 – Evolution du rendu des personnages humains dans les films d'animation Pixar. Toy Story utilise une représentation par NURBS alors que les deux autres utilisent des surfaces de subdivision. L'attention doit principalement être portée sur la capacité d'expression des visages et le rendu des mains des différents personnages.

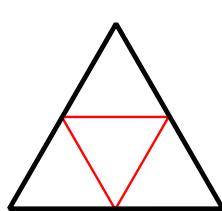
Au cours de cette période, les progrès ne sont pas seulement liés à la création de schémas de subdivision avec de nouvelles propriétés mais également à l'étude de ceux-ci. Stam propose une extension des travaux d'Halstead *et al.* [HKD93] pour calculer efficacement tout point de la surface limite de Catmull-Clark [Sta98a] et de Loop [Sta98b]. Zorin et Schröder trouvent une méthode pour construire le schéma de subdivision associé à une surface tensorielle B-Spline uniforme de degré quelconque [ZS01].

### 3.1.5 Classification des schémas de subdivision

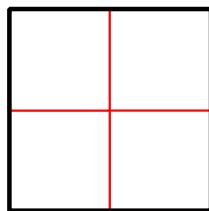
Avec autant de schémas de subdivision ayant des propriétés parfois communes et parfois distinctes, une classification s'avère nécessaire. En 2000, Zorin propose son *subdivision zoo* [Zor00] qui reste à ce jour, et malgré l'apparition de nouveaux schémas, la classification la plus courante lorsque l'on parle de subdivision **uniforme** (*i.e.* même règle pour tout le maillage) et **stationnaire** (*i.e.* même règle pour chaque itération). Cette classification concerne un ensemble de propriétés qui sont détaillées ci-dessous.

#### Type de maillages

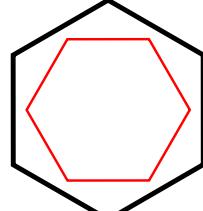
Le type de maillage d'un schéma de subdivision est le type d'une face régulière de ce schéma. Par exemple, le schéma de Catmull-Clark est quadrilatéral et celui de Loop est triangulaire. Seules ces deux catégories existaient à l'origine mais par la suite d'autres ont été ajoutées : hexagonale et hybride (faces de différents types). Différents exemples sont présentés en Figure 3.3.



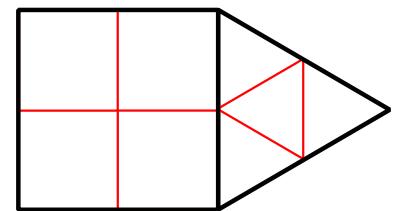
Triangle (Loop)



Quadrangle (Catmull-Clark)



Hexagonal (Honeycomb)



Hybride (Stam-Loop)

FIGURE 3.3 – Exemples de la subdivision topologique de faces régulières de plusieurs schémas. Les faces originales sont en noir et les faces obtenues par subdivision en rouge.

### Interpolant ou approximant

Un schéma est considéré comme **interpolant** si les sommets du maillage de contrôle appartiennent toujours à la surface limite. Ainsi, dans un schéma **interpolant**, les sommets sont reportés d'une itération à l'autre. A l'inverse, pour un schéma **approximant**, les sommets sont déplacés en fonction de leurs voisins lors d'une itération. Un exemple de schéma approximant et un exemple de schéma interpolant sont donnés en Figure 3.4.

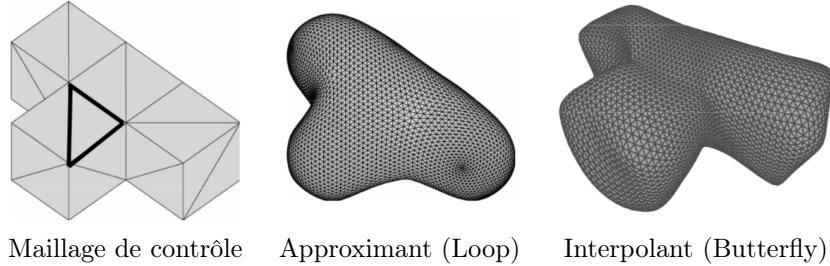


FIGURE 3.4 – Comparaison entre un schéma approximant et un schéma interpolant appliqués sur le même maillage de contrôle. Les images sont extraites de [Zor00].

### Dualité

Un schéma interpolant est forcément primal mais un schéma approximant peut être primal ou dual. **Un schéma primal subdivise les faces** : une face devient plusieurs faces de même type (*e.g.* triangles, quadrangles...). A l'inverse, **un schéma dual subdivise les sommets** : chaque sommet, arête, et face devient une face. Dans le cas des schémas primaux, toutes les faces générées sont de même type et les sommets conservent leur valence. Dans le cas des schémas duals, tous les sommets générés sont de même valence et les faces conservent leur type. Les exemples courants sont Catmull-Clark [CC78] pour les schémas primaux et Doo-Sabin [DS78] pour les schémas duals (voir Figure 3.5).

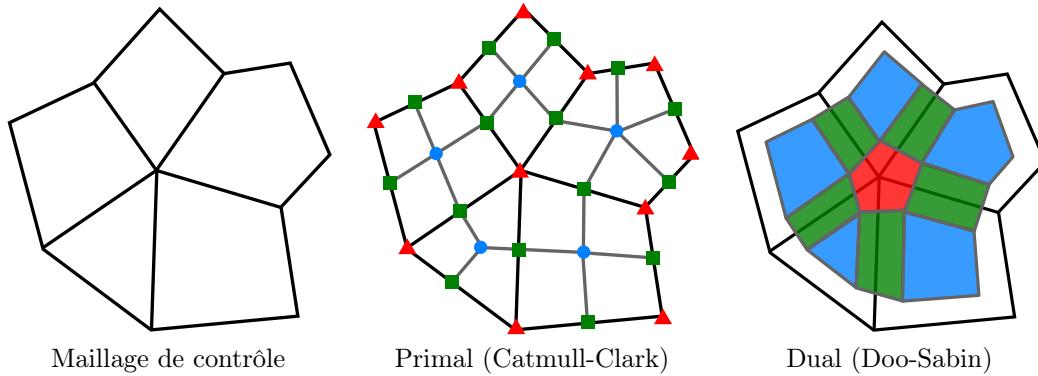


FIGURE 3.5 – Exemples d'une itération sur un schéma primal et un schéma dual sur le même maillage de contrôle quelconque. Le schéma primal génère un **sommet** par sommet, par arête, et par face (respectivement représentés par des triangles rouges, des carrés verts, et des ronds cyans). Le schéma dual, quant à lui, génère une **face** par sommet (en rouge), par arêtes (en vert), et par face (en cyan). Remarquez que pour le schéma de Catmull-Clark, toutes les faces générées sont quadrangulaires alors que pour le schéma de Doo-Sabin, tous les sommets générés sont de valence 4.

### Continuité de la surface limite

Lorsque le schéma de subdivision est appliqué une infinité de fois, une surface limite est obtenue et la continuité de celle-ci peut être étudiée. Cette continuité est double : celle dans le cas des maillages réguliers est distinguée de celle autour des sommets irréguliers. Par exemple, la surface limite obtenue par application du schéma de Catmull-Clark est  $C_2$ -continue dans les zones régulières et  $C_1$ -continue autour des sommets extraordinaires. Par abus de langage, cette continuité est souvent appelée **continuité du schéma** plutôt que continuité de la surface limite.

Halstead *et al.* définissent les matrices de subdivision [HKD93] qui permettent de calculer l'évolution du voisinage d'un sommet du maillage au cours des itérations. En étudiant le vecteur propre associé à la valeur propre  $\lambda_1 = 1$  de ces matrices de subdivision, ils calculent la position du sommet (quelle que soit la valence de celui-ci) sur la surface limite en fonction de son voisinage. Ulrich Reif démontre que la continuité d'un schéma de subdivision peut se faire par l'analyse des valeurs/vecteurs propres des matrices de subdivision [Rei95, RS00]. Les conditions sont les suivantes, où les valeurs propres  $\lambda_i$  sont triées par module décroissant :  $|\lambda_i| \geq |\lambda_{(i+1)}| > 0$ , et se cumulent :

- Si  $|\lambda_1| = 1 > |\lambda_2|$  alors le schéma converge (point-fixe unique)
- Si  $|\lambda_2| = |\lambda_3| > |\lambda_4| = |\lambda_5|$  alors le schéma est  $C_1$ -continu

Pour les continuités d'ordre supérieur, il faut étudier précisément les cartes caractéristiques [Rei95, RS00].

Un exemple comparatif entre les surfaces limites obtenues par deux schémas de continuités différentes appliqués sur le même maillage de contrôle est donnée en [Figure 3.6](#).

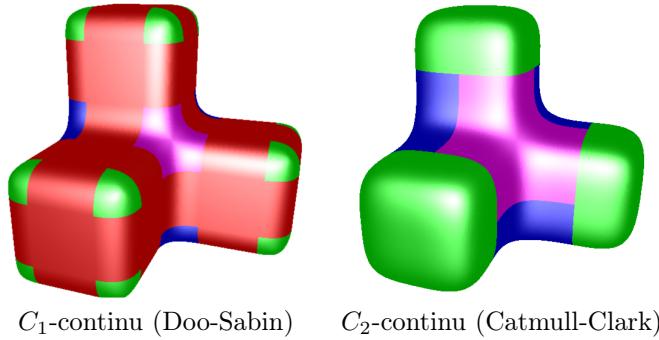


FIGURE 3.6 – Exemples de surfaces limites obtenues à partir du même maillage de contrôle mais de deux schémas de continuités différentes. Remarquez que les arêtes obtenues par le schéma de Doo-Sabin sont plus vives que celles obtenues par le schéma de Catmull-Clark.

## 3.2 Le schéma de subdivision de Catmull-Clark

Edwin Catmull et James "Jim" Clark [CC78] introduisent, en même temps que la notion de schéma de subdivision, leur schéma éponyme qui est une extension des surfaces tensorielles B-Splines bi-cubiques uniformes aux maillages quelconques. Ce schéma primal et approximant produit des surfaces  $C_2$ -continues ( $C_1$ -continues autour des sommets extraordinaires) à partir de règles de subdivision simples ; c'est pourquoi il reste encore aujourd'hui le plus courant, avec le schéma de Loop, malgré l'existence de plusieurs schémas techniquement plus performants.

### 3.2.1 Formulation sous forme de masques

La formulation traditionnelle se fait sous forme de masque qui nous donnent l'influence d'un voisinage de sommets (sous forme d'une combinaison barycentrique) sur un sommet de la prochaine itération. Ces masques dépendent à la fois du type de sommet créé et du voisinage qui peut être irrégulier dans certains cas. Les différents types de sommets sont :

- les **F-sommets** (*faces*) qui sont le centre de gravité d'une face ;
- les **E-sommets** (*edges*) qui dépendent des sommets des deux faces adjacentes à une arête ;
- les **V-sommets** (*vertices*) qui sont influencés en grande partie par un sommet central puis de manière moindre par les sommets des faces incidentes à ce sommet.

Chaque face est découpée en autant de faces quadrangles qu'elle possède de sommets. Pour cela, le V-sommet issu d'un sommet est relié par une arête aux deux E-sommets générés par les arêtes incidentes à ce sommet, et ces deux E-sommets sont reliés par une arête au F-sommet de la face (voir [Figure 3.7](#)).

Pour chaque type de sommet il y aura distinction entre les cas réguliers (*i.e.* lorsque le maillage est de type grille : toutes les faces quadrangles et tous les sommets de valence 4) et irréguliers (tous les autres cas). Les formules sont détaillées ci-dessous et les masques sont schématisés dans la [Figure 3.8](#).

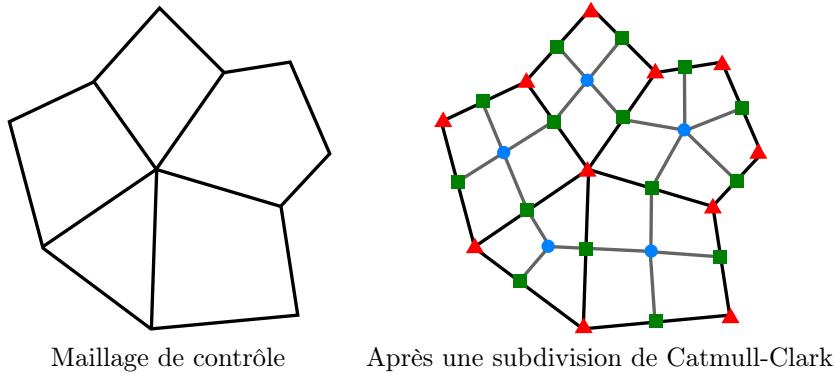


FIGURE 3.7 – Exemple d'une subdivision de Catmull-Clark sur un maillage quelconque. Les V-sommets sont représentés par des triangles rouges, les E-sommets par des carrés verts, et les F-sommets par des ronds cyans. Remarquez que toutes les faces ainsi générées sont quadrangles.

### Les F-sommets

Le F-sommet d'une face est simplement l'isobarycentre (ou centre de gravité) de la face. Chaque sommet  $f_i$  de la face a la même influence sur le F-sommet :  $1/4$  dans le cas régulier (face quadrangle) et  $1/s$  dans le cas d'une face à  $s$  sommets notés  $f_0 \dots f_{s-1}$  :

$$F = \sum_{i=0}^{s-1} \frac{f_i}{s}$$

### Les E-sommets

Les E-sommets doivent être calculés après les F-sommets des faces incidentes à l'arête car ceux-ci interviennent dans le calcul. Le E-sommet d'une arête est l'isobarycentre des deux sommets  $e_0$  et  $e_1$  de l'arête et des deux F-sommets des faces incidentes  $F_0$  et  $F_1$  :

$$E = \frac{e_0 + e_1 + F_0 + F_1}{4}$$

Dans le cas régulier (les deux faces incidentes sont quadrangles), le E-sommet est généralement directement calculé à partir des deux sommets de l'arête  $e_0$  et  $e_1$  et des deux sommets  $f_{i,0}$  et  $f_{i,1}$  de chaque face  $i \in \{0; 1\}$  :

$$E = \frac{6(e_0 + e_1) + (f_{0,0} + f_{0,1} + f_{1,0} + f_{1,1})}{16}$$

### Les V-sommets

Pour les V-sommets il faut faire la distinction entre 3 cas : le cas régulier (le sommet central est de valence 4 et toutes les faces incidentes sont quadrangles), le cas irrégulier où toutes les faces incidentes sont quadrangles, et le cas irrégulier où au moins une face n'est pas un quadrangle. Dans le cas d'un maillage de type grille, les poids du sommet central  $v$ , de ses sommets voisins  $e_0 \dots e_3$  et des sommets avec lesquels il partage une face mais pas une arête  $f_0 \dots f_3$  sont respectivement  $36/64$ ,  $6/64$ , et  $1/64$  :

$$V = \frac{36v + 6(e_0 + e_1 + e_2 + e_3) + (f_0 + f_1 + f_2 + f_3)}{64}$$

Dans le cas où toutes les faces adjacentes au sommet central sont quadrangles mais que ce sommet est de valence  $k$ , la formule est la suivante :

$$V = \frac{(4k^2 - 7k)v + \sum_{i=0}^{k-1} (6e_i + f_i)}{4k^2}$$

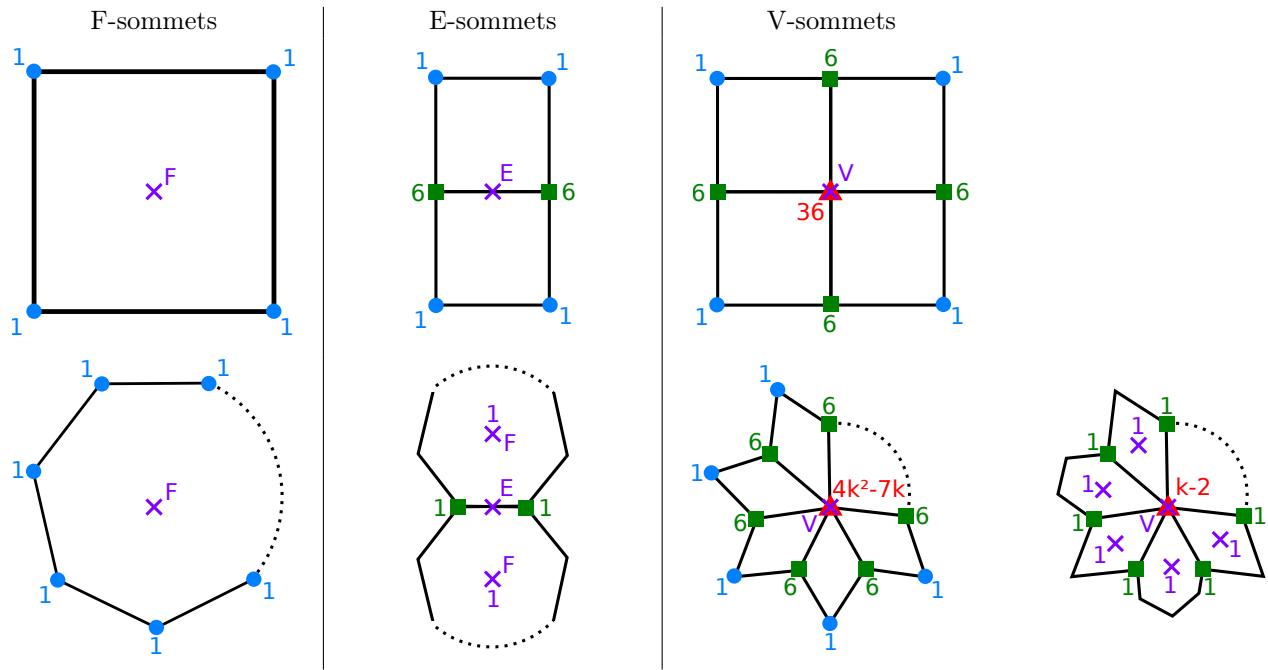


FIGURE 3.8 – Les masques de subdivision du schéma de Catmull-Clark. Les sommets calculés sont représentés par les croix violettes, les sommets qui influent sur celui-ci sont représentés respectivement par un rond cyan lorsqu'ils ont une face commune (voisinage 2), par un carré vert lorsqu'ils ont une arête en commun (voisinage 1) et par un triangle rouge si il s'agit du même sommet. Les poids d'un masque ont tous le même dénominateur commun qui n'est pas indiqué sur les schémas pour des raisons de lisibilité.

Dans le cas où les faces adjacentes ne seraient pas des quadrangles la formule est changée de la manière suivante pour faire intervenir les F-sommets des faces incidentes  $F_0 \dots F_{k-1}$  :

$$V = \frac{(k-2)v + \sum_{i=0}^{k-1} \frac{e_i + F_i}{k}}{k}$$

Même si les règles de subdivision prennent en compte le cas des faces non-quadrangles, le schéma de Catmull-Clark ne crée que des faces quadrangles (en reliant un V-sommet aux deux E-sommets des arêtes incidentes d'une même face ainsi qu'au F-sommet de cette face). Ainsi, toute face à  $s$  sommets génère dès la première itération  $s$  faces quadrangles. Il est donc usuel de contraindre le maillage de contrôle à n'être composé que de faces quadrangles, chacune étant subdivisée en quatre nouvelles faces à chaque itération. Cette convention sera respectée par la suite.

### 3.2.2 Formulation sous forme de patches

Dans le cas des schémas approximants, chaque itération ajoute des sommets et déplace les anciens, ce qui crée de l'instabilité (les sommets d'une itération donnée ne sont pas à la même position à l'itération suivante). Pour pallier ce problème, la notion de surface limite (*i.e.* la surface obtenue après une infinité théorique d'itérations) est abordée, d'abord par Halstead [HKD93], puis par Stam [Sta98a]. Il est intéressant de noter que la formulation de Stam (optimisée par la suite pour les cartes graphiques) est encore couramment utilisée pour les applications de rendu temps réel de surface de subdivision.

#### La méthode d'Halstead

Halstead [HKD93] part du constat qu'un voisinage en étoile composé d'un sommet central et de l'ensemble des sommets des faces incidentes à ce sommet est suffisant pour calculer un voisinage de l'itération suivante, topologiquement similaire et centré sur le même sommet (voir Figure 3.9). Ceci implique que la matrice qui transforme un voisinage en un voisinage identique de l'itération suivante est carrée, elle peut donc être à nouveau appliquée sur le nouveau voisinage. Cette matrice de transformation  $\mathcal{M}(k)$ , de taille  $(2k+1) \times (2k+1)$  (où  $k$  désigne la valence du sommet central), et la combinaison barycentrique  $B_k$  associée au point fixe sont également données en Figure 3.9.

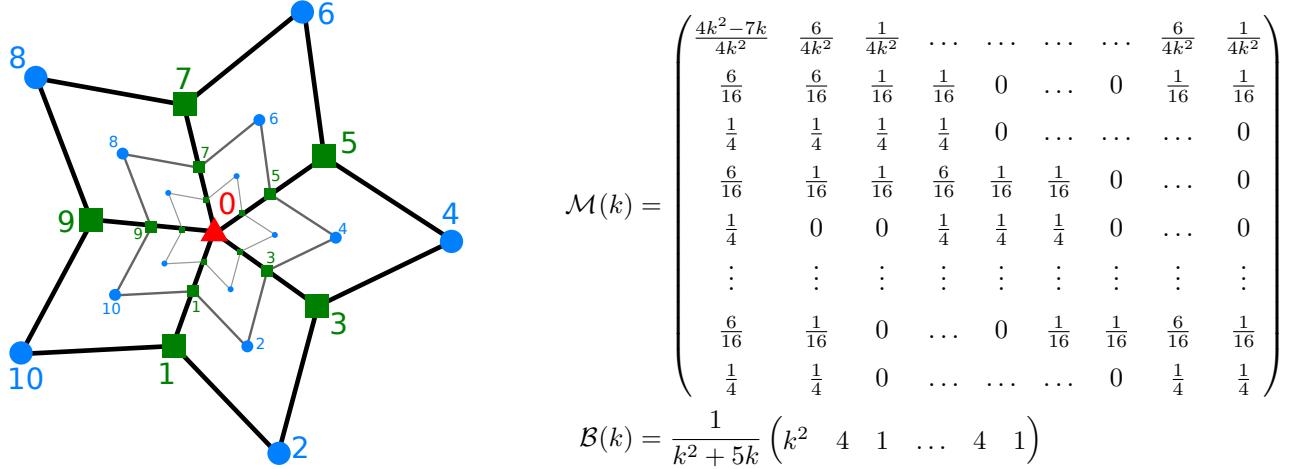


FIGURE 3.9 – A gauche, deux itérations du schéma de Catmull-Clark sur une étoile composée d'un sommet central (triangle bleu) et de l'ensemble des sommets de ses faces incidentes. La topologie de l'étoile du voisinage est strictement la même au cours des itérations mais la géométrie converge vers un point unique qui appartient à la surface limite. A droite, la matrice qui transforme un voisinage en étoile en celui de l'itération suivante et la combinaison barycentrique qui projette le sommet central sur la surface limite.

Cette matrice étant issue de combinaisons barycentriques convexes, elle est stochastique (*i.e.* tous ses coefficients sont positifs et la somme de chaque ligne est égale à 1). De plus, cette matrice possède une valeur propre unique maximale égale à 1. Comme démontré dans la [Sous-section 2.3.3](#), ces propriétés de la matrice implique qu'il existe un unique vecteur propre à gauche  $\mathcal{B}(k)$ , associé à la valeur propre 1, qui décrit une combinaison barycentrique (*i.e.* vecteur dont la somme des coefficients égale à 1). Cette combinaison barycentrique, qui permet de calculer directement la position de ce sommet après une infinité d'application de la matrice sur le voisinage et donc sa position sur la surface limite, est donnée en [Figure 3.9](#).

Halstead propose simplement l'algorithme suivant : la subdivision de Catmull-Clark est appliquée autant de fois que nécessaire pour obtenir le nombre de sommets/faces désirés puis chaque sommet du maillage est projeté sur la surface limite en appliquant sur chacun la combinaison barycentrique  $\mathcal{B}(k)$ .

### La méthode de Stam

Stam [[Sta98a](#)], quant à lui, utilise la propriété des surfaces de Catmull-Clark de se comporter comme des surfaces tensorielles B-Splines bi-cubiques uniformes dans les voisinages de type grille. La méthode de Stam n'est appliquée que sur un maillage dont toutes les faces sont quadrangles et ne possèdent pas plus d'un sommet de valence irrégulière (différente de 4). Pour s'assurer qu'aucune face n'ait plusieurs sommets extraordinaire, une itération de Catmull-Clark est appliquée sur le maillage. En effet, une face quadrangle possédant plusieurs sommets extraordinaire est subdivisée en quatre faces qui possèdent au maximum un sommet extraordinaire. Un exemple est donné en [Figure 3.10](#).

Une fois que le maillage respecte les conditions de départ, il est décomposé en patches (un par face du maillage) qui seront traités indépendamment les uns des autres. Un patch est composé d'une face centrale et d'un anneau de faces adjacentes à celle-ci. Ce patch est le voisinage nécessaire et suffisant pour calculer un carreau de surface limite indexé par des coordonnées locales de surface  $(u; v) \in [0; 1] \times [0; 1]$ . Les patches sont décomposés en deux catégories : réguliers et irréguliers ; un patch régulier est un patch de type grille alors qu'un patch irrégulier est un patch dont la face centrale possède un (et un seul) sommet de valence irrégulière (voir exemples en [Figure 3.10](#)). Chaque patch régulier ainsi extrait correspond exactement à un maillage de contrôle d'une surface tensorielle B-Spline bi-c cubique ; tout point  $(u; v)$  du carreau de surface limite correspondant est calculé de la même manière que dans le cas des surfaces B-Splines. Les patches irréguliers sont décomposés en quatre sous-patches par l'application des règles de

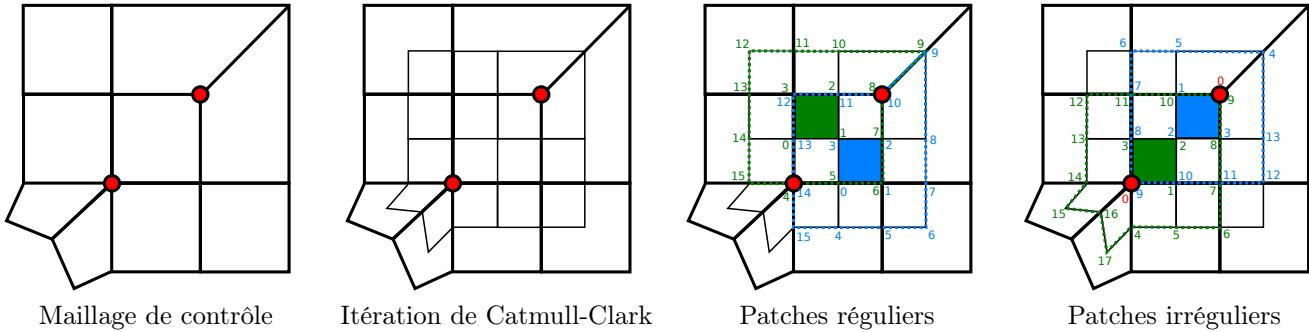


FIGURE 3.10 – Schéma illustrant la génération préalable de patches pour la méthode de Stam. Une itération des règles de Catmull-Clark est d’abord effectuée pour s’assurer que plusieurs sommets extraordinaire (représentés par des ronds rouges) ne partagent pas la même faces. Ensuite les patches réguliers et irréguliers sont extraits. La numérotation des sommets des patches sera toujours la suivante : 0, 1, 2, et 3 pour ceux de la face centrale puis ceux de l’anneau extérieur en respectant l’ordre trigonométrique. Dans le cas des patches irréguliers, le sommet extraordinaire est systématiquement numéroté à 0.

subdivision de Catmull-Clark (voir Figure 3.11). Parmi ces sous-patches, trois sont réguliers et sont donc traités comme des surfaces B-Splines. Le dernier est topologiquement identique au patch parent et doit être à nouveau décomposé.

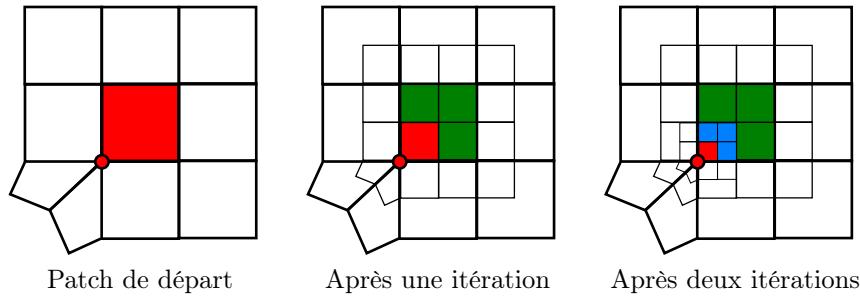


FIGURE 3.11 – Résultat d’une succession d’itérations sur un patch irrégulier indiqué en rouge. La face centrale de chaque patch permettant le calcul d’une surface tensorielle B-Spline uniforme bi-cubique est colorée en vert et en cyan selon le nombre d’itérations nécessaires à sa création. Le patch irrégulier est répété à chaque étape mais subit une transformation contractante.

Soient les transformations  $\mathcal{T}_0 \dots \mathcal{T}_3$  qui associent, dans l’espace géométrique, un patch parent à un de ses sous-patches. Dans le cas des patches réguliers, les transformations sont notées dans l’ordre trigonométrique et la position de la première n’a pas d’importance. Dans le cas des patches irréguliers,  $\mathcal{T}_0$  associe le patch au sous-patch irrégulier et les transformations  $\mathcal{T}_1, \mathcal{T}_2$ , et  $\mathcal{T}_3$  aux sous-patches réguliers (voir Figure 3.12).

Les matrices de transformations associées  $\mathcal{M}_0 \dots \mathcal{M}_3$  utilisent les coefficients suivants (où  $k$  est la valence du sommet extraordinaire s’il existe, et sinon 4) :

$$V = \frac{36}{64} ; E = \frac{6}{64} ; F = \frac{1}{64} ; A = \frac{1}{4} ; B = \frac{6}{16} ; C = \frac{1}{16} ; v = \frac{4k^2 - 7k}{4k^2} ; e = \frac{6}{4k^2} ; f = \frac{1}{4k^2}$$

Ces coefficients sont ceux du schéma de Catmull-Clark (dans le cas où toutes les faces du maillage sont quadrangles) et également ceux utilisés dans la méthode d’Halstead. La matrice  $\mathcal{M}_0(k)$  est donnée à titre d’exemple pour les cas  $k = 3$  et  $k \geq 4$  (les coefficients en noir sont fixes, ceux en rouge dépendent de la valence, ceux nuls sont ignorés pour plus de lisibilité).

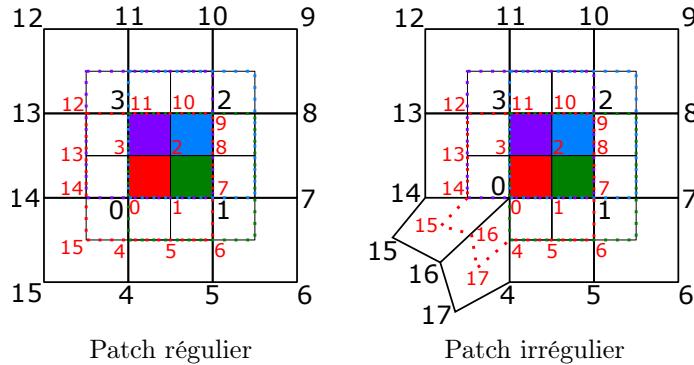


FIGURE 3.12 – Décomposition d'un patch de Catmull-Clark régulier et d'un patch irrégulier en quatre sous-patches. Les sous-patches sont indiqués en pointillés et la face centrale de chacun est représentée par un carré plein. Les couleurs rouge, vert, cyan, et violet indiquent respectivement les sous-patches obtenus par les transformations  $\mathcal{T}_0$ ,  $\mathcal{T}_1$ ,  $\mathcal{T}_2$ , et  $\mathcal{T}_3$ .

$$\mathcal{M}_0(3) = \left( \begin{array}{cccc|cc|c} v & e & f & e & e & f & & f \\ B & B & C & C & C & C & & \\ A & A & A & A & & & & \\ B & C & C & B & C & & & C \\ \hline B & C & C & & B & C & & C \\ A & A & & & A & A & & \\ C & B & & & C & B & C & \\ E & V & E & F & F & E & F & F \\ \hline C & B & B & C & C & C & C & \\ F & E & V & E & F & E & F & F \\ C & C & B & B & & C & C & \\ E & F & E & V & F & F & E & F \\ C & & B & C & & C & C & B \\ A & & A & A & & & & A \end{array} \right)$$
  

$$\mathcal{M}_0(k \geq 4) = \left( \begin{array}{cccc|cc|c|ccccc} v & e & f & e & e & f & & f & e & f & \cdots & e & f \\ B & B & C & C & C & C & & & & & & & \\ A & A & A & A & & & & & & & & & \\ B & C & C & B & & & & & C & C & & & \\ \hline B & C & & & B & C & & & & & & C & C \\ A & A & & & A & A & & & & & & & \\ C & B & & & C & B & C & C & & & & & \\ E & V & E & F & F & E & F & E & & & & & \\ \hline C & B & B & C & C & C & C & & & & & & \\ F & E & V & E & F & E & F & E & & & & & \\ C & C & B & B & & C & C & & & & & & \\ E & F & E & V & F & E & F & E & & & & & \\ \hline C & & B & C & & C & C & & C & C & & & \\ A & & A & A & & & & & A & A & & & \\ \vdots & & & & & & & & \ddots & A & A & A & \ddots \\ B & & & & C & & & & C & C & B & C & C \\ \vdots & & & & A & & & & \ddots & A & A & A & \ddots \\ B & & & & C & & & & C & C & B & C & C \\ A & & & & A & & & & \ddots & A & A & A & \ddots \end{array} \right)$$

Comme la matrice  $\mathcal{M}_0(k)$  est carrée (de taille  $(2k + 8) \times (2k + 8)$ ) la combinaison barycentrique  $\mathcal{B}_0(k)$  est également calculée :

$$\mathcal{B}_0(k) = \frac{1}{k^2 + 5k} ( \begin{array}{cccccccccccccc} k^2 & 4 & 1 & 4 & 4 & 1 & 0 & \dots & 0 & 1 & 4 & \dots & 1 & 4 & 1 \end{array} )$$

En plus d'associer dans l'espace géométrique un patch à un de ses sous-patchs, les transformations  $\mathcal{T}_0 \dots \mathcal{T}_3$  associent également, dans l'espace des textures, le carré unitaire à un de ses quatre sous-carrés de la manière présentée en [Figure 3.13](#). Pour calculer, dans un carreau de surface correspondant à un patch irrégulier, le point de coordonnées locales  $(u; v) \in [0 ; 1] \times [0 ; 1]$ , l'algorithme récursif suivant est appliqué :

- Selon les valeurs de  $(u; v)$ , la transformation  $\mathcal{T}_i$  est sélectionnée
- Les coordonnées  $(u; v)$  sont transformées en  $(u'; v')$  selon  $\mathcal{T}_i^{-1}$  dans l'espace de texture
- La matrice  $\mathcal{M}_i(k)$  est appliquée sur le patch dans l'espace géométrique
- Si  $i = 0$ , alors l'algorithme recommence pour le nouveau patch irrégulier et les coordonnées  $(u'; v')$
- Sinon, le nouveau patch est devenu régulier et le point  $(u'; v')$  est calculé par la méthode classique des B-Splines bi-cubiques uniformes.

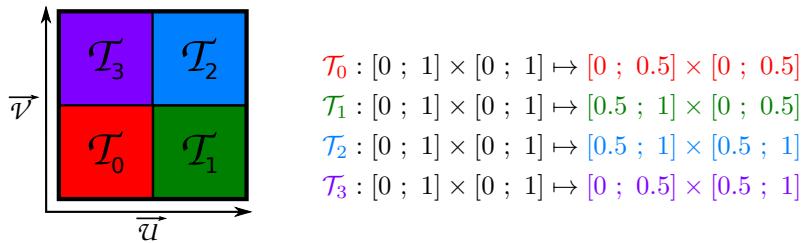


FIGURE 3.13 – Application des transformations  $\mathcal{T}_i$  dans l'espace des textures.

Le seul cas qui n'est pas géré par l'algorithme est le cas du point de coordonnées de textures  $(0;0)$  qui est répété à l'identique à chaque itération. Celui-ci est directement calculé par la méthode d'Halstead, avec la combinaison  $\mathcal{B}_0(k)$ . Ainsi, le carreau de surface limite qui correspond à un patch irrégulier est également calculable en tout point.

### 3.2.3 Formulation sous forme d'un automate CIFS

Par sa méthode, Stam donne le point de départ de la construction d'un automate CIFS qui correspond au schéma de subdivision de Catmull-Clark. L'algorithme de Stam, qui agit dans l'espace géométrique, est décomposé pour obtenir que chaque transformation soit à la fois définie dans l'espace des textures pour générer des adresses à partir des coordonnées  $(u; v)$  du carreau correspondant, et dans l'espace barycentrique à partir des matrices de Stam.

#### Génération des adresses

Soient  $\mathcal{T}_0 \dots \mathcal{T}_3$  les transformations définies par Stam. Dans l'espace des textures, chacune de ces transformations associe le carré unitaire à l'un de ses sous-carrés. Soient également  $\mathcal{P}_0 \dots \mathcal{P}_3$  les transformations dites "point-fixes" qui associent tout point de l'espace barycentrique au point fixe de  $T_i$  (*i.e.*  $\mathcal{P}_i = \mathcal{T}_i^\infty$ ).

Tout point  $(u; v)$  du carré unitaire est indexable par une suite de transformations (potentiellement infinie) qui est appelée adresse du point. Pour s'assurer que l'adresse soit finie, et donc l'algorithme exécutable en temps fini, seuls les points dont l'adresse est de la forme  $\mathcal{T}_a \mathcal{T}_b \dots \mathcal{T}_y \mathcal{P}_z$  sont calculés. L'algorithme utilisé pour convertir un de ces points en une adresse est donné en [Figure 3.14](#).

La liste des transformations de l'adresse permet, dans l'espace des textures, d'associer le carré unitaire au point  $(u; v)$ . L'exploitation de cette adresse pour le calcul de la surface limite est expliquée dans les sections suivantes.

**FONCTION** conversion\_coords\_vers\_adresse (u, v)

    adresse  $\leftarrow \emptyset$

**BOUCLE INFINIE**

        // Calcul du numéro de la transformation

        ur  $\leftarrow \text{round}(u)$

        vr  $\leftarrow \text{round}(v)$

        i  $\leftarrow 2.vr + (ur+vr) \% 2$

        // Ajout de la transformation à l'adresse

**SI** u = ur **ET** v = vr **ALORS**

            RENOVIE adresse.push\_back( $\mathcal{P}_i$ )

**SINON**

            adresse.push\_back( $\mathcal{T}_i$ )

            u  $\leftarrow 2(u-0.5.ur)$

            v  $\leftarrow 2(v-0.5.vr)$

**FIN SI**

**FIN BOUCLE INFINIE**

**FIN FONCTION**

FIGURE 3.14 – Algorithme de génération des adresses pour le schéma de Catmull-Clark.

### Construction de l'IFS barycentrique des cas réguliers

Dans l'espace géométrique, les transformations  $\mathcal{T}_i$  définies par Stam associent un patch à l'un de ses sous-patches. Dans le cas régulier, ces transformations sont définies par des matrices carrés stochastiques  $\mathcal{M}_i(4)$ , qui peuvent donc également agir dans l'espace barycentrique BI<sup>16</sup>. Un IFS barycentrique basé sur ces matrices peut donc être défini. L'attracteur de cet IFS, dans l'espace barycentrique correspond lorsqu'il est plongé dans l'espace géométrique à la surface limite du patch obtenue par une infinité de subdivisions. Chaque adresse  $\mathcal{T}_a\mathcal{T}_b\dots\mathcal{T}_y\mathcal{P}_z$  définie ci-dessus correspond au point de l'attracteur qui est la combinaison barycentrique  $\mathcal{B}_z\mathcal{M}_y\dots\mathcal{M}_b\mathcal{M}_a$ . La combinaison ainsi calculée est le point de l'attracteur de coordonnées (u; v) qui sont les coordonnées qui ont permis la construction de l'adresse.

Cette combinaison est ensuite projetée sur un patch en appliquant les pondérations de la combinaison sur les points de contrôle du patch. Le point ainsi obtenu est le point de coordonnées locales (u; v) du carreau de surface limite correspondant au patch. De la même manière que [ZT96, SLC05], en générant une tessellation de l'attracteur dans l'espace barycentrique, puis en projetant chaque combinaison de cette tessellation sur le même patch, la même tessellation est obtenue, mais cette fois sur la surface limite du patch. En projetant la tessellation sur l'ensemble des patches du maillage, la surface limite est obtenue.

La méthode de Stam nécessite une séquence de transformations représentées par des matrices pour calculer la surface limite d'un patch. Pour calculer un point de la surface limite, il faut appliquer des produits matriciels jusqu'à ce que le maillage devienne régulier puis calculer le point limite comme celui d'une B-Spline uniforme. A l'inverse, dans la méthode présentée ici, il suffit d'appliquer une combinaison barycentrique sur le patch de contrôle pour obtenir directement le point sur la surface limite.

### Gestion des patches irréguliers par un automate CIFS

Comme dit précédemment, les CIFS permettent de construire des IFS dont les transformations changent la nature de l'objet. C'est le cas dans la méthode de Stam où les transformations  $\mathcal{T}_i, i \in \{1, 2, 3\}$  associent un patch irrégulier (de  $(2k+8)$  sommets) à un patch régulier (de 16 sommets). Un automate dont les états sont naturellement "patch régulier" et "patch irrégulier" et dont les transitions correspondent aux transformations  $\mathcal{T}_j$  est construit. Ensuite, en étudiant les transformations  $\mathcal{T}_j$  qui ne modifient pas la forme du patch (donc les transitions qui bouclent sur un même état), les transformations  $\mathcal{P}_j$ , qui associent un patch à un point de la surface limite, sont calculées. L'état "point de la surface limite" (équivalent à l'état "point de l'attracteur") et les transitions correspondant aux transformations  $\mathcal{P}$  sont ajoutées à l'automate (voir Figure 3.15).

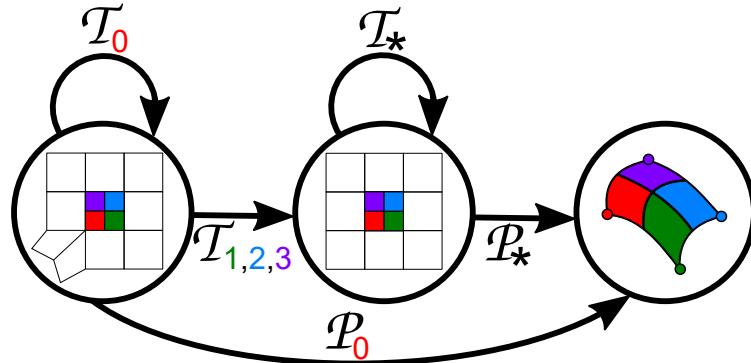


FIGURE 3.15 – Automate CIFS du schéma de Catmull-Clark. Les indices  $\star$  correspondent à n'importe quel numéro.

### 3.3 Méthode de génération d'automate CIFS pour les schémas de subdivision

En s'inspirant de la génération de l'automate CIFS correspondant au schéma de Catmull-Clark, une méthode permettant de trouver l'automate CIFS d'un schéma de subdivision uniforme à support compact (*i.e.* dont la zone d'influence de chaque point de contrôle n'est pas trop grande) a été défini. Cet automate permet ensuite de générer de manière efficace une tessellation de la surface limite à partir d'un certain nombre de combinaisons barycentriques à appliquer sur le maillage de contrôle.

#### 3.3.1 Etape 1 : création des patches

Dans cette étape, l'objectif est de trouver la forme d'un patch régulier, puis d'un irrégulier. Pour cela, un maillage régulier pour le schéma est généré, puis subdivisé une ou plusieurs fois en suivant les règles du schéma. Il faut alors trouver l'ensemble le plus compact de point du maillage original qui est répété plusieurs fois à l'étape suivante et dont les nouveaux sommets sont calculés uniquement à partir des anciens. Quelques règles trouvées empiriquement peuvent aider dans la recherche de ce patch :

- Si le schéma est primal, le patch est centré sur une face ;
- Si le schéma est dual, le patch est centré sur un sommet ;
- Le patch doit respecter une invariance par rotation centrale (90° pour un carré, 120° pour un triangle) ;
- Si le schéma entraîne une rotation du maillage, il peut être nécessaire de l'appliquer plusieurs fois avant de chercher une auto-similarité de topologie.

Une fois le patch régulier trouvé, le patch irrégulier est construit en changeant la valence d'un sommet pour les schémas primaux ou le nombre d'arêtes d'une face pour les schémas duals. Cette méthode de construction des patches irréguliers implique que les maillages de contrôle doivent respecter les conditions suivantes :

- Si le schéma est primal, le maillage de contrôle n'a que des faces régulières et chaque patch a au maximum un seul sommet extraordinaire
- Si le schéma est dual, le maillage de contrôle n'a que des sommets de valence régulière et chaque patch a au maximum une face extraordinaire.

Lorsqu'un maillage est modélisé dans le but que sa surface limite soit calculée par un CIFS, ces contraintes sont respectées lors de la construction. Si le maillage n'a pas été modélisé dans ce but et qu'il ne respecte pas les conditions, il est subdivisé en suivant les règles habituelles du schéma jusqu'à ce qu'il soit correct. Comme les schémas de subdivision créent uniquement de la régularité et par conséquent isolent les cas extraordinaire, le maillage finira toujours par respecter ces conditions.

### 3.3.2 Etape 2 : génération des transformations, calcul des points fixes, et construction de l'automate

En partant du patch irrégulier et des sous-patches issus de celui-ci, une transformation est définie par sous-patches. S'il n'y a qu'une seule transformation qui associe un patch irrégulier parent à un patch irrégulier enfant, elle est appelée  $\mathcal{T}_0$ , et les autres transformations sont numérotées dans l'ordre trigonométrique. La structure de l'automate est premièrement définie avec les états "patch régulier" et "patch irrégulier" ainsi que les transitions correspondantes (*N.B.* : pour certains schémas, il peut y avoir plusieurs cas d'irrégularités et dans ce cas plusieurs états "patch irrégulier"). L'état final "point de la surface limite" (qui est l'autre nom de l'état "point de l'attracteur" dans le cas des surfaces de subdivision) est ajouté à l'automate et chaque transition  $\mathcal{T}_i$  qui boucle sur un état génère une transition étiquetée  $\mathcal{P}_i$  qui relie cet état à l'état final.

Les matrices stochastiques  $\mathcal{M}_i$  correspondant aux transformations  $\mathcal{T}_i$  sont construites à partir des règles de subdivision où chaque ligne de la matrice correspond à la combinaison barycentrique qui permet le calcul d'un nouveau point en fonction des anciens. Toutes les matrices associées à des transformations qui conservent la topologie sont carrées, les combinaisons barycentriques qui correspondent aux point-fixes, tangentes, et courbures sont calculées par analyse des vecteurs propres. De cette manière, toutes les transformations correspondant à des transitions de l'automate sont définies dans l'espace barycentrique.

### 3.3.3 Etape 3 : génération des adresses

Maintenant que l'automate est correctement construit, il faut définir les mots qu'il va lire : les adresses. Pour cela, la première chose à faire est de définir la forme de l'espace des paramètres. Celle-ci correspond à la forme du morceau de surface limite obtenu à partir d'un patch unique. D'après les différents schémas testés, la règle empirique suivante se dessine :

- si le schéma est primal, l'espace des paramètres a la forme d'une face régulière
- Si le schéma est dual, l'espace a la forme du dual d'une face régulière

Cette règle vient du fait que le morceau de surface limite correspondant à un patch a autant de morceaux de surface limite voisins que le patch a de patches voisins. Dans le cas où le patch est centré sur une face (primal), le patch a autant de voisins que sa face centrale a d'arêtes et le morceau de surface limite correspondant autant de bords. Dans le cas où le patch est centré sur un sommet, le nombre de voisins du patch est la valence du sommet central et donc cette valence est le nombre de bord du morceau de surface limite.

Par exemple, pour les schémas primaux de Catmull-Clark et de Loop, il s'agit du carré unitaire  $((u; v) \in [0; 1]^2)$  et du triangle équilatéral barycentrique convexe unitaire  $((u; v; w) \in [0; 1]^3 / (u + v + w = 1)$ . Pour les schémas duals de Doo-Sabin et *Honeycomb*, il s'agit du dual du carré (le carré unitaire) et du dual de l'hexagone (le triangle équilatéral unitaire). Cette règle s'explique par le fait que le voisinage des patches est transposé sur le voisinage des morceaux de surface limite. Si le schéma est primal, chacun de ses patches est centré sur une face et les patches voisins sont ceux centrés sur les faces voisines. Pour conserver le voisinage, chaque arête de la face centrale génère un bord du morceau de surface limite : la forme de la face centrale est conservée. Si le schéma est dual, chacun de ses patches est centré sur un sommet et les patches voisins sont ceux centrés sur les sommets voisins. Pour conserver le voisinage, chaque arête issue du sommet central génère un bord du morceau de surface limite : la forme du morceau de surface limite est celle du dual d'une face régulière du maillage.

Une fois l'espace des paramètres défini, celui-ci est découpé en autant de morceaux que de transformations. Ces morceaux sont identiques entre-eux et ne se superposent pas, ils sont par contre joints bord à bord pour s'assurer que l'ensemble des paramètres soient définis.

Dans les sections suivantes, différents schémas de subdivision à support compact sont présentés sous forme de règles de subdivision, puis la construction de l'automate CIFS correspondant est décrit selon la méthode ci-dessus :

- la topologie d'un patch régulier et irrégulier est trouvée et les sous-patches sont donnés,
- la structure de l'automate est déduite de la liste des transformations,
- la matrice/combinaison barycentrique correspondant à la transformation associée à chaque transition est construite/calculée,
- l'espace des paramètres est déduit de la forme du patch, il est découpé en autant de morceaux que de transformations  $\mathcal{T}_i$  et l'algorithme de génération des adresses correspondant est donné.

Des schémas de subdivision à support non-compact (*i.e.* taille du patch trop conséquente) ou hybrides (*i.e.* ayant plusieurs types de faces régulières et donc de voisinages différents) sont également étudiés et les problèmes qu'ils impliquent sont discutés.

## 3.4 Le schéma de Doo-Sabin

Publiant leur article en même temps que celui de Catmull et Clark [CC78], Daniel Doo et Malcolm Sabin introduisent un autre schéma de subdivision éponyme [DS78]. Ce schéma est l'exemple le plus courant de schéma dual et produit des surfaces tensorielles B-Splines bi-quadratiques uniformes  $C_1$ -continues.

### 3.4.1 Formulation sous forme de masques

Le schéma de Doo-Sabin se décrit par l'algorithme suivant (voir Figure 3.16) :

- pour chaque sommet de chaque face, un nouveau sommet est généré ;
- tous les nouveaux sommets issus de la même ancienne face constituent une nouvelle face (les **F-faces**) ;
- tous les sommets issus de la même ancienne arête constituent également une nouvelle face (les **E-faces**) ;
- tous les sommets issus du même ancien sommet constituent encore une nouvelle face (les **V-faces**).

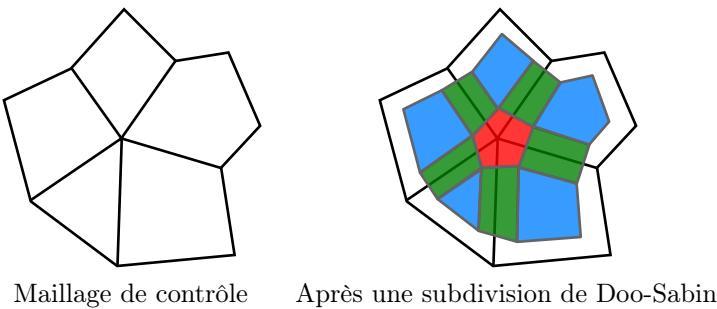


FIGURE 3.16 – Exemple d'une subdivision de Doo-Sabin sur un maillage de topologie quelconque. Les F-faces sont représentées en cyan, les E-faces en vert et les V-faces en rouge. Remarquez que tous les sommets générés sont de valence 4.

Pour une face, un nouveau sommet est généré par ancien sommet et ceux-ci sont associés. Tous les anciens sommets de la face influent les nouveaux sommets, l'ancien sommet (dit principal) correspond à l'influence la plus grande, puis l'influence de chaque ancien sommet décroît avec l'éloignement de cet ancien sommet du sommet principal. Un exemple de cas régulier (face quadrangle) et un exemple irrégulier sont donnés en Figure 3.17

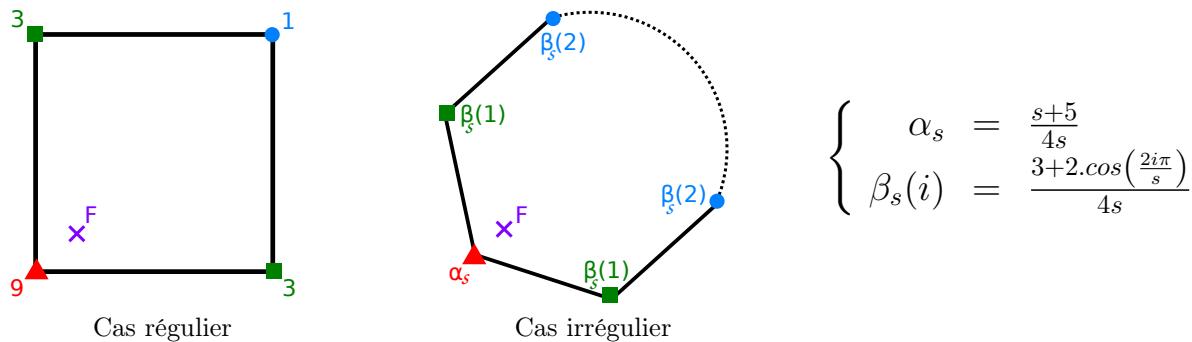


FIGURE 3.17 – Les masques du schéma de Doo-Sabin dans les cas régulier (à gauche) et irrégulier (au centre). Les coefficients du cas irréguliers sont renseignés à droite où  $s$  désigne le nombre de sommets de la face,  $\alpha_s$  l'influence du sommet principal (*i.e.* l'ancien sommet qui correspond au nouveau sommet) et  $\beta_s(i)$  est une fonction qui calcule l'influence d'un autre sommet de la face en fonction du nombre de sommets  $i \in [1; \lfloor (s-1)/2 \rfloor]$  (y compris lui-même) qui le sépare du sommet principal.

### 3.4.2 Formulation sous forme d'un automate CIFS

Pour formuler le schéma sous forme d'un automate CIFS, les conditions imposées au maillage sont :

- tous les sommets sont de valence 4
- deux faces non-quadrangles ne peuvent être adjacentes

Pour s'assurer que le maillage est conforme, il suffit d'appliquer deux fois la subdivision classique de Doo-Sabin : la première itération rend tous les sommets de valence 4 et la seconde itération sépare les faces non-quadrangles qui partagent un sommet ou une arête.

Les patches de Doo-Sabin sont composés d'un sommet central et de ses quatre faces incidentes. Les patches réguliers ne comportent que des faces quadrangles et donc 9 sommets. Les irréguliers ont au maximum une seule face non-quadrangle dont le nombre de sommets est  $s$ , et donc le patch est de taille  $(s + 5)$ . Ces patches sont décomposés en quatre sous-patches (voir Figure 3.18), le CIFS possède donc quatre transformations. L'automate peut alors être construit (voir Figure 3.19).

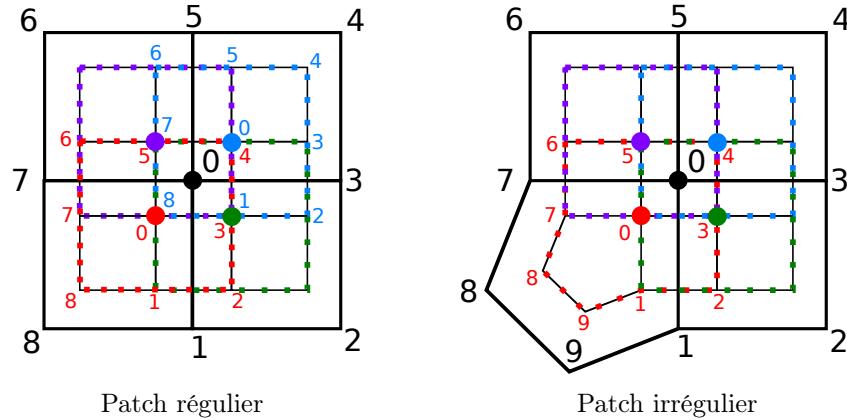


FIGURE 3.18 – Deux exemples de patches de Doo-Sabin : régulier et irrégulier (une face pentagonale) centrés sur un sommet (rond noir). Les sous-patches associés à chacun sont représentés en rouge, vert, cyan et violet (le sommet central par un rond et le contour du sous-patches par des pointillés). Le patch régulier est décomposé en quatre sous-patches réguliers et le patch irrégulier en trois sous-patches réguliers et un patch irrégulier (celui du côté de la face irrégulière, représenté en rouge), qui est identique au patch parent.

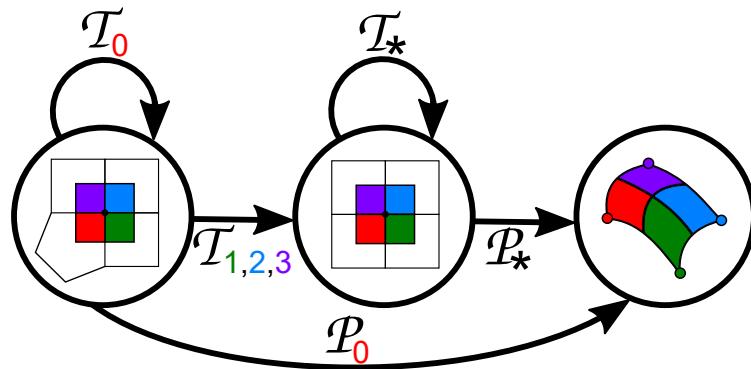


FIGURE 3.19 – Automate CIFS du schéma de Doo-Sabin. Les indices  $*$  correspondent à n'importe quel indice de transformation.

Deux exemples de matrices de transformation sont donnés ci-dessous :

$$\mathcal{M}_0(3) = \begin{pmatrix} \alpha_3 & \beta_3(1) & & & \beta_3(1) \\ \beta_3(1) & \alpha_3 & & & \beta_3(1) \\ \hline 3/16 & 9/16 & 3/16 & 1/16 & \\ 9/16 & 3/16 & 1/16 & 3/16 & \\ 9/16 & & 3/16 & 1/16 & 3/16 \\ 9/16 & & & 3/16 & 1/16 & 3/16 \\ 3/16 & & & & 1/16 & 3/16 & 9/16 \\ \hline \beta_3(1) & \beta_3(1) & & & & & \alpha_3 \end{pmatrix}$$

$$\mathcal{M}_0(5) = \begin{pmatrix} \alpha_5 & \beta_5(1) & & & \beta_5(1) & \beta_5(2) & \beta_5(2) \\ \beta_5(1) & \alpha_5 & & & \beta_5(2) & \beta_5(2) & \beta_5(1) \\ \hline 3/16 & 9/16 & 3/16 & 1/16 & & & \\ 9/16 & 3/16 & 1/16 & 3/16 & & & \\ 9/16 & & 3/16 & 1/16 & 3/16 & & \\ 9/16 & & & 3/16 & 1/16 & 3/16 & \\ 3/16 & & & & 1/16 & 3/16 & 9/16 \\ \hline \beta_5(1) & \beta_5(2) & & & \alpha_5 & \beta_5(1) & \beta_5(2) \\ \beta_5(2) & \beta_5(2) & & & \beta_5(1) & \alpha_5 & \beta_5(1) \\ \beta_5(2) & \beta_5(1) & & & \beta_5(2) & \beta_5(1) & \alpha_5 \end{pmatrix}$$

Dans cette matrice, les cinq lignes centrales sont fixes quel que soit le nombre de sommet de la face irrégulière (seul le nombre de zéros finaux change). Les lignes supérieures sont toujours au nombre de 2 et les lignes inférieures au nombre de  $(s-2)$ . Ces lignes comportent toujours un  $\alpha$  et  $(s-1)\beta(i)$  (avec deux  $\beta$  par valeurs de  $i \in [1; \lfloor (s-1)/2 \rfloor]$  sauf pour le  $i$  le plus grand dans le cas pair).

Le schéma de Doo-Sabin est dual et ces faces régulières sont des quadrangles. D'après la règle énoncée dans la méthode générique, l'espace des paramètres a la forme du dual d'une face régulière. Dans le cas présent, le dual d'un quadrangle est un quadrangle, l'espace des paramètres est donc le carré unitaire. Et comme un patch devient quatre sous-patches, ce carré unitaire est découpé en quatre morceaux identiques qui sont les quatre sous-carrés. Donc les adresses sont construites pour le schéma de Doo-Sabin comme pour celui de Catmull-Clark.

## 3.5 Le schéma *Simplest* ou *Mid-edge*

En 1996, Jörg Peters et Ulrich Reif proposent un schéma de subdivision quadrangle, dual, et  $C_1$ -continu (comme celui de Doo-Sabin) qu'ils appellent *Simplest* (le plus simple). Ce schéma est considéré comme le plus simple possible car il ne possède qu'une seule règle de subdivision qui relie les milieux de deux arêtes adjacentes de la même face en une nouvelle arête. Ce schéma est souvent appelé dans la communauté comme le schéma *Mid-edge* (milieu d'arête) en référence à cette règle de construction.

### 3.5.1 Formulation sous forme de masques

La formulation sous forme de masques est directe et évidente : un sommet est ajouté au milieu de chaque arête. Les masques qui correspondent à deux itérations du schéma sont plus intéressants (voir Figure 3.20).

En observant les masques de double-itération, il apparaît que le schéma *Mid-edge* est semblable au schéma de Doo-Sabin (seuls les coefficients changent). Ceci n'est pas vraiment étonnant vu que les deux schémas ont les mêmes propriétés :

- schémas quadrangulaires
- schémas duaux
- schémas  $C_1$ -continus

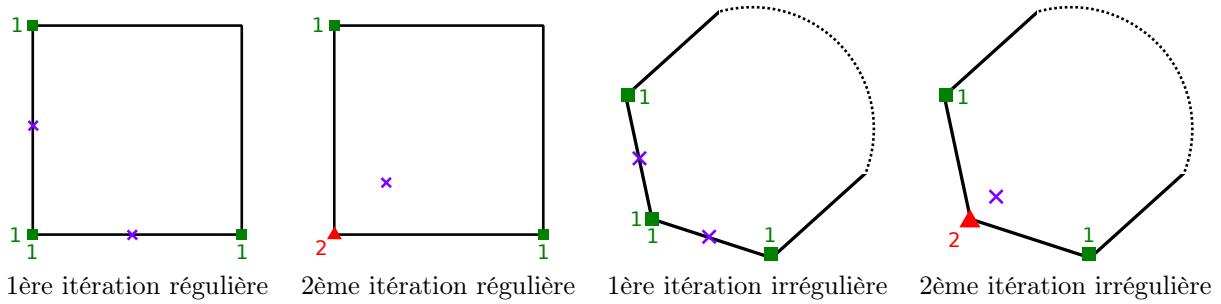


FIGURE 3.20 – Les masques réguliers et irréguliers du schéma *Mid-edge* en une et deux itérations.

### 3.5.2 Formulation sous-forme d'un automate CIFS

Comme les règles de subdivision du schéma *Mid-edge* appliquées deux fois sont similaires aux règles de subdivision de Doo-Sabin, il est logique que les patches des deux schémas soient de topologie identiques (voir Figure 3.21) et par conséquent les automates CIFS ont la même structure (cf. Figure 3.22).

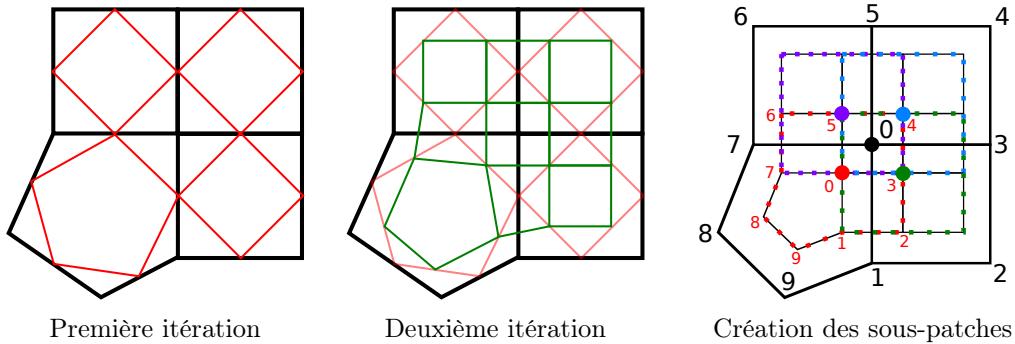


FIGURE 3.21 – Crédit des sous-patches d'un patch irrégulier du schéma *Mid-edge*. La topologie du patch et celles ses sous-patches sont identiques à celles du schéma de Doo-Sabin

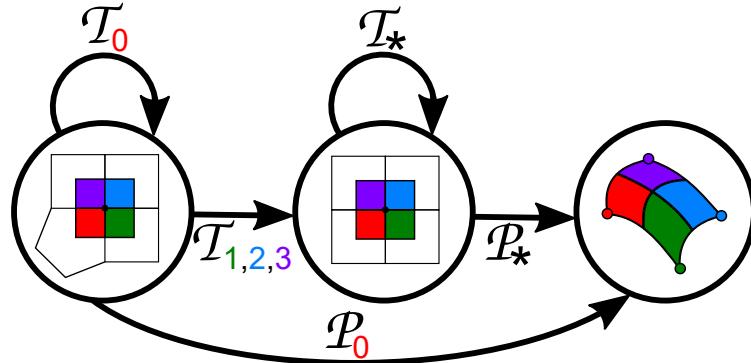


FIGURE 3.22 – Automate CIFS du schéma *Mid-edge*. Les indices  $*$  correspondent à n'importe quel numéro de transformation.

La seule différence entre les deux schémas sont les coefficients des règles. Ainsi, la seule différence entre les deux automates CIFS sont les matrices de transformation qui sont donc modifiées de la manière suivante :

$$\mathcal{M}_0(5) = \begin{pmatrix} 2/4 & 1/4 & & & 1/4 \\ 1/4 & 2/4 & & & \rightarrow & 1/4 \\ 1/4 & 2/4 & 1/4 & & & \\ 2/4 & 1/4 & & 1/4 & & \\ 2/4 & & 1/4 & 1/4 & & \\ 2/4 & & & 1/4 & 1/4 & \\ 1/4 & & & & 1/4 & 2/4 \\ 1/4 & & & & & 2/4 & 1/4 \\ & & & & & 1/4 & 2/4 & 1/4 \\ & & & & & 1/4 & 2/4 \\ & & & & & & 1/4 & 2/4 \\ & & & & & & & 1/4 & 2/4 \end{pmatrix}$$

La partie haute de la matrice est fixe à part le coefficient rouge de la deuxième ligne qui est toujours sur la dernière colonne. Chaque ligne  $i$  de la partie basse a pour  $i$ -ème coefficient  $2/4$  encadré par deux coefficients à  $1/4$  (modulo la dernière colonne qui boucle sur la deuxième).

## 3.6 Le schéma de Loop

En 1987, Charles Teorell Loop [Loo87] introduit le premier schéma de subdivision adapté aux maillages triangulaires. En effet, les deux schémas précédents [CC78, DS78] étaient principalement prévus pour des maillages quadrangulaires. Ce schéma primal et approximant se base sur la notion de Box-Spline pour générer une surface  $C_2$ -continue dans le cas régulier du voisinage d'un sommet de valence 6. Les coefficients des cas irréguliers sont ensuite calculés de manière à ce que le schéma produise une surface  $C_1$ -continue autour des sommets extraordinaire.

### 3.6.1 Formulation sous forme de masques

Une itération du schéma de Loop crée deux types de sommets :

- Les E-sommets qui sont insérés pour chaque arête,
- Les V-sommets qui sont des anciens sommets déplacés selon leurs voisins.

Après la création de ses sommets, chaque E-sommet est relié aux deux V-sommets issus des sommets de l'arête et aux quatre E-sommets issus des faces incidentes à l'arête. Ainsi tous les E-sommets générés sont de valence 6 et les V-sommets conservent la topologie de leur voisinage. Un exemple est donné en Figure 3.23.

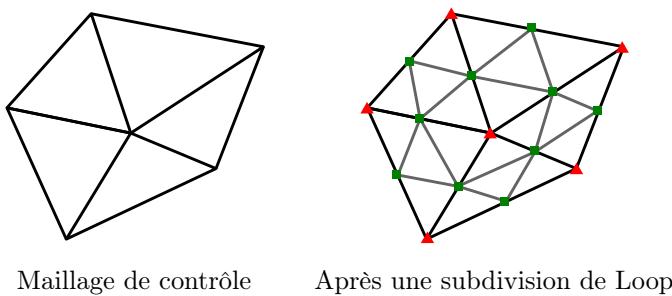


FIGURE 3.23 – Exemple d'une subdivision du schéma de Loop sur un maillage triangulaire quelconque. Les V-sommets sont représentés par des triangles rouges et les E-sommets par des carrés verts. Remarquez que tous les E-sommets générés sont de valence 6 et que les V-sommets conservent leur topologie de voisinage.

La formulation des Box-Splines pour les maillages triangulaires donne les coefficients des masques pour les arêtes et les sommets réguliers. Ceci implique que le schéma produit une surface  $C_2$ -continue pour une grille triangulaire régulière. Dans le cas des sommets de valence irrégulière (différente de 6), l'objectif est de générer une surface  $C_1$ -continue. Tout d'abord, la règle de subdivision des arêtes est conservée car les arêtes ont toujours deux faces incidentes dans un maillage fermé. Pour la règle de déplacement des sommets, Loop définit que l'ancien sommet a une pondération de  $\alpha$  et ses  $k$  voisins une influence de  $\beta$ , ce qui implique que  $\alpha + k\beta = 1$ . Les masques de subdivision sont donnés en Figure 3.24.

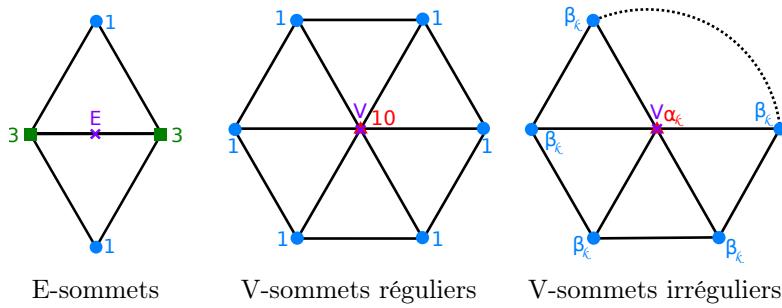


FIGURE 3.24 – Masques du schéma de subdivision de Loop. Les E-sommets générés et les V-Sommets déplacés sont représentés par des croix violettes. Le sommet central est représenté par un triangle rouge, les sommets arêtes par des carrés verts, et les sommets face par des ronds cyans. Le dénominateur commun des influences n'est pas précisé pour des raisons de lisibilité.

Par analyse de Fourier, Loop propose les coefficients suivants :

$$\begin{cases} \alpha_k &= \frac{3}{8} + \left(\frac{3}{8} + \frac{1}{4}\cos\left(\frac{2\pi}{k}\right)\right)^2 \\ \beta_k &= \frac{1}{k} \left[\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\left(\frac{2\pi}{k}\right)\right)^2\right] \end{cases}$$

Quelques années plus tard, Warren propose une formulation simplifiée des coefficients permettant d'éviter les fonctions trigonométriques [War95](page 94) :

$$\text{Si } k = 3 \quad \begin{cases} \alpha_k &= \frac{7}{16} \\ \beta_k &= \frac{3}{16} \end{cases} \quad \text{Si } k > 3 \quad \begin{cases} \alpha_k &= \frac{5}{8} \\ \beta_k &= \frac{3}{8k} \end{cases}$$

### 3.6.2 Formulation sous forme d'un automate CIFS

Les patches de Loop, donnés par Stam dans [Sta98b], sont composés d'une face centrale et de l'anneau de faces l'entourant. Les conditions sur le maillage de départ sont les suivantes :

- toutes les faces sont triangulaires,
- aucune face ne possède plus d'un sommet extraordinaire.

Le schéma de Loop est primal, une face triangulaire génère quatre faces triangulaires, les sommets conservent la topologie de leur voisinage d'une itération à l'autre et tous les nouveaux sommets sont de valence 6. Le patch de Loop régulier et un exemple de patch irrégulier sont donnés en Figure 3.25 et l'automate CIFS, similaire au deux précédents, est donné en Figure 3.26.

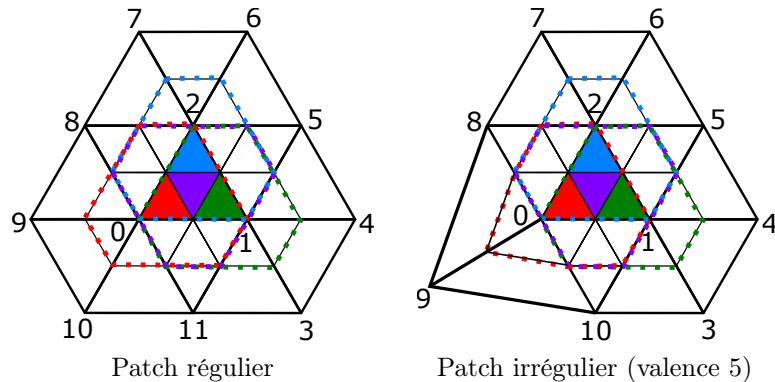


FIGURE 3.25 – Décomposition d'un patch régulier et d'un patch irrégulier de Loop en quatre sous-patches par application des règles de subdivision. Remarquez que les sous-patches vert, cyan, et violet sont réguliers alors que le sous-patch rouge est de la même configuration topologique que le patch parent.

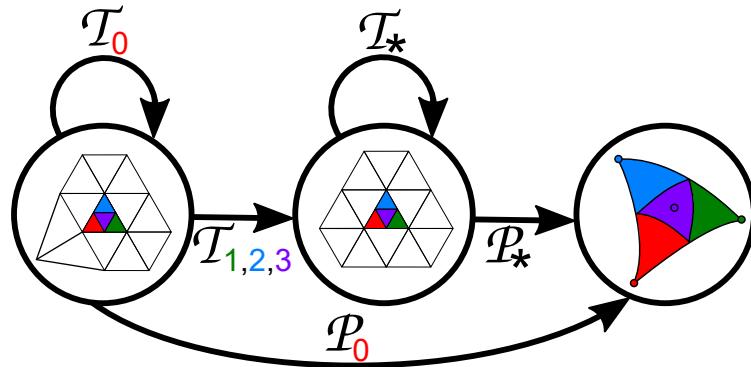


FIGURE 3.26 – Automate CIFS du schéma de Loop. Les indices  $\star$  correspondent à n'importe quel numéro de transformation.

La matrice de taille  $(k + 6) \times (k + 6)$  associées à la transformation  $T_0$  ainsi que les combinaisons barycentriques associées au point-fixe  $P_0$  sont données ci-dessous :

$$\mathcal{M}_0(k) = \left( \begin{array}{ccccccccc|ccc} \alpha_k & \beta_k & \beta_k & & & \beta_k & \dots & \dots & \dots & \beta_k \\ \hline 3/8 & 3/8 & 1/8 & & & 1/8 & & & & \rightarrow 1/8 \\ 3/8 & 1/8 & 3/8 & & & & & & & \rightarrow 3/8 \\ 1/8 & 3/8 & 1/8 & & & & & & & \rightarrow 1/16 \\ 1/16 & 10/16 & 1/16 & 1/16 & 1/16 & 1/16 & & & & \\ 1/8 & 3/8 & 3/8 & & & 1/8 & & & & \\ 1/16 & 1/16 & 10/16 & & & 1/16 & 1/16 & 1/16 & 1/16 & \\ 1/8 & & 3/8 & & & & 1/8 & 3/8 & & \\ 3/8 & & 1/8 & & & & 3/8 & 1/8 & & \\ \hline 3/8 & & & & & 1/8 & 3/8 & 1/8 & & \\ \vdots & & & & & \ddots & \ddots & \ddots & & \\ 3/8 & & & & & & 1/8 & 3/8 & 1/8 & \\ 3/8 & 1/8 & & & & & & 1/8 & 3/8 & \\ \end{array} \right)$$

Les coefficients de la première ligne dépendent de la valence  $k$  du sommet extraordinaire. Celui de la première colonne sera toujours  $\alpha_k$ , ceux des deux suivantes, ainsi que des  $(k - 2)$  dernières, égaux à  $\beta_k$ . Les lignes centrales sont fixes hormis les coefficients marqués en rouge qui sont toujours sur la dernière colonne. Les lignes du bas ont toutes comme premier coefficient  $3/8$ . Ensuite, chaque ligne  $i$  a pour  $i$ -ème coefficient  $3/8$  et pour  $(i - 1)$ -ème et  $(i + 1)$ -ème coefficient  $1/8$  (avec une boucle sur la dernière ligne).

Le schéma de Loop est un schéma primal dont les faces régulières sont des triangles. Ceci implique que l'espace des paramètres de Loop est celui utilisé pour plaquer des textures sur un triangle : le triangle équilatéral unitaire barycentrique. Ce triangle est défini comme l'ensemble des paramètres  $(u; v; w) \in [0; 1]^3$  tel que  $u + v + w = 1$ . Chaque patch est décomposé en quatre sous-patches, l'espace des paramètres est donc découpé en quatre morceaux identiques. Ces morceaux sont les sous-triangles équilatéraux définis dans la Figure 3.27 et les adresses sont construites par l'algorithme donné en Figure 3.28.

### 3.7 Le schéma *Honeycomb*

Akelman et Srinivasan introduisent en 2002 un schéma pour les maillages hexagonaux qu'ils appellent *Honeycomb* en référence aux alvéoles des ruches à miel [AS02]. Les schémas de Catmull-Clark et de Doo-Sabin sont duals (le premier génère des faces quadrangulaires, le second des sommets de valence 4) la question d'un schéma dual au schéma  $\sqrt{3}$  [Kob00] s'est donc posée. Comme celui-ci génère des faces triangulaires et, dans le cas régulier uniquement, des sommets de valence 6, il faut trouver un schéma qui génère des sommets de valence 3 et dans son cas régulier des faces hexagonales.

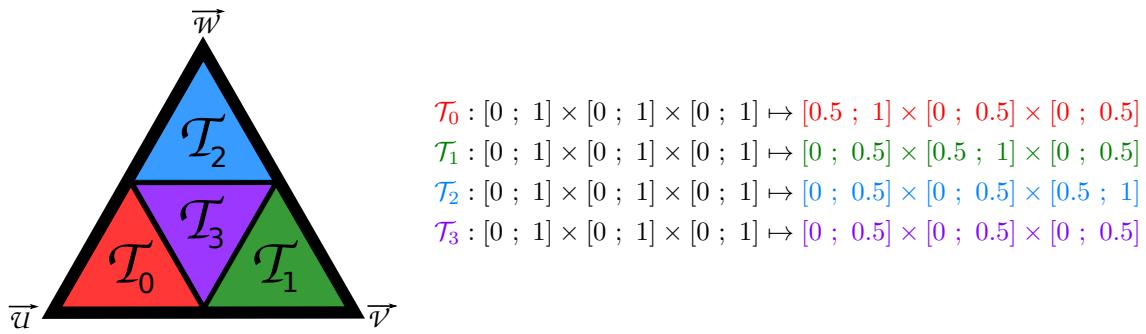


FIGURE 3.27 – Paramétrisation des morceaux de surface du schéma de Loop. Les transformations  $\mathcal{T}_0$ ,  $\mathcal{T}_1$ , et  $\mathcal{T}_2$  sont de simples homothéties mais la transformation  $\mathcal{T}_3$  subit forcément une rotation.

**FONCTION** conversion\_coords\_vers\_adresse (u, v, w)

adresse  $\leftarrow \emptyset$

**BOUCLE INFINIE**

// Si un point-fixe est atteint

**SI** u = 1 **RENOIE** adresse.push\_back( $\mathcal{P}_0$ )

**SI** v = 1 **RENOIE** adresse.push\_back( $\mathcal{P}_1$ )

**SI** w = 1 **RENOIE** adresse.push\_back( $\mathcal{P}_2$ )

// Sinon

**SI** u > 0.5 **ALORS**

adresse.push\_back( $\mathcal{T}_0$ )

u  $\leftarrow$  2.u-1

v  $\leftarrow$  2.v

w  $\leftarrow$  2.w

**SINON SI** v > 0.5 **ALORS**

adresse.push\_back( $\mathcal{T}_1$ )

u  $\leftarrow$  2.u

v  $\leftarrow$  2.v-1

w  $\leftarrow$  2.w

**SINON SI** w > 0.5 **ALORS**

adresse.push\_back( $\mathcal{T}_2$ )

u  $\leftarrow$  2.u

v  $\leftarrow$  2.v

w  $\leftarrow$  2.w-1

**SINON**

adresse.push\_back( $\mathcal{T}_3$ )

u  $\leftarrow$  u+v-w

v  $\leftarrow$  v+w-u

w  $\leftarrow$  u+w-v

**FIN SI**

**FIN BOUCLE INFINIE**

**FIN FONCTION**

FIGURE 3.28 – Algorithme de génération des adresses pour le schéma de Loop.

L'intérêt principal des maillages hexagonaux est que les hexagones réguliers composent le pavage régulier optimal du plan (*i.e.* dont les tuiles ont le meilleur ratio aire/périmètre). Cette solution optimale, trouvée par sélection naturelle et utilisée depuis par les abeilles pour stocker le miel, fait l'objet du **Théorème du nid d'abeille** démontré par Thomas Hales [Hal01].

### 3.7.1 Formulation sous forme de masques

A l'intérieur de chaque face, un nouveau sommet est généré par arête de cette face et tous les sommets de la face participent au calcul de tous les nouveaux sommets de la face. Ensuite, tous les nouveaux sommets issus de la même face forment une nouvelle face (F-face) et les deux sommets associés à la même arête mais dans deux faces différentes sont reliés par une arête ce qui à pour conséquence de construire des faces (V-faces) autour des anciens sommets. Un schéma est donné en Figure 3.29

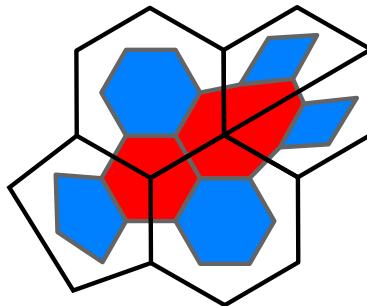


FIGURE 3.29 – Exemple d'une subdivision du schéma *Honeycomb* sur un maillage quelconque. Les V-faces sont représentées en rouge et les F-faces en cyan. Remarquez que chaque F-face a le même nombre de sommets que la face dont elle est issue et que tous les nouveaux sommets sont de valence 3.

Les règles de subdivision du schéma *Honeycomb* sont à effectuer en deux étapes. La première consiste à insérer un sommet-arête au milieu de chaque arête. Ensuite, d'une manière similaire au schéma de Doo-Sabin, tous les sommets-arêtes d'une face participent au calcul du nouveau sommet associé à chaque arête. Les masques de subdivision sont donnés en Figure 3.30.

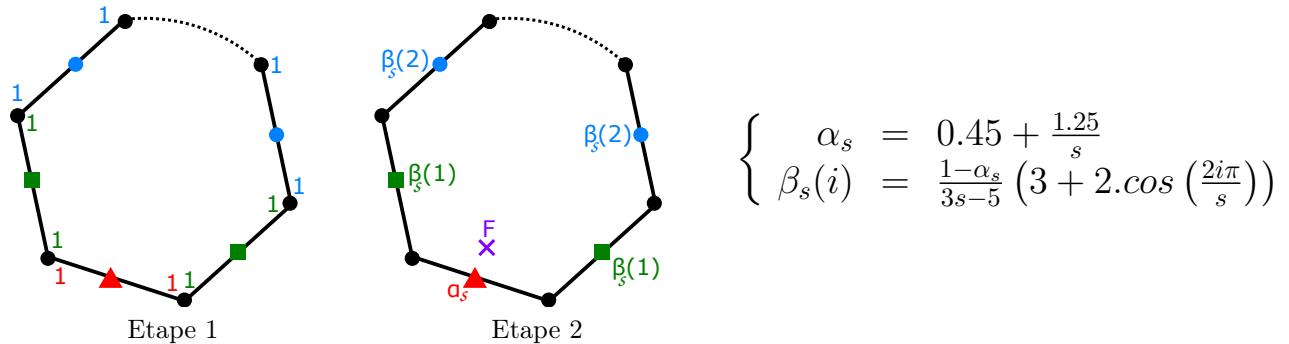
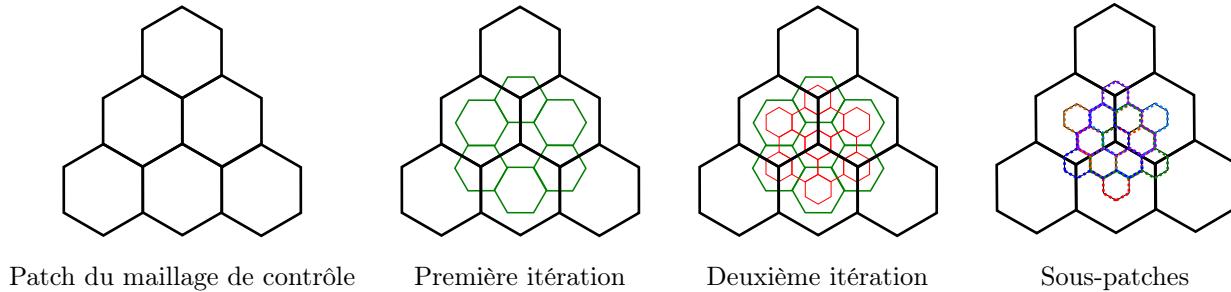


FIGURE 3.30 – Les deux masques consécutifs du schéma *Honeycomb*. La première étape génère un sommet-arête au milieu de chaque arête et la seconde utilise ces sommets-arêtes pour le calcul des nouveaux sommets. Les coefficients de la deuxième étape sont renseignés à droite où  $s$  désigne le nombre de sommets de la face,  $\alpha_s$  l'influence du sommet-arête principal (*i.e.* celui qui est associé à l'arête qui génère un sommet) et  $\beta_s(i)$  est une fonction qui calcule l'influence d'un autre sommet-arête de la face en fonction du nombre de sommets  $i \in [1; \lfloor (s-1)/2 \rfloor]$  (y compris lui-même) qui le séparent du sommet-arête principal.

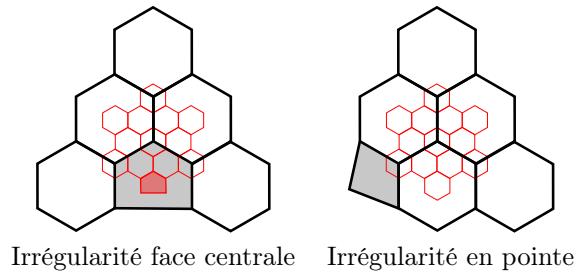
### 3.7.2 Formulation sous forme d'un automate CIFS

Le schéma *Honeycomb* est un schéma dual, son patch est donc centré sur un sommet. De plus, tout comme le schéma *Mid-edge*, c'est un schéma pivotant (*e.g.* un schéma dual qui construit les nouvelles faces en fonction des arêtes et non des sommets), il faut donc deux étapes de subdivision pour trouver une auto-similarité de patch. Le patch trouvé à partir de ces indices est donné en [Figure 3.31](#).



**FIGURE 3.31 –** Création des sous-patches à partir d'un patch régulier du schéma *Honeycomb*. Deux itérations du schéma sont nécessaires pour obtenir une auto-similarité.

Il faut ensuite trouver les patches irréguliers. Comme le schéma est dual, il conserve la topologie des faces et génère des sommets de valence fixe (ici de valence 3), ce qui signifie que les irrégularités sont le nombre de sommets des faces. En imposant qu'un patch ne peut contenir qu'une seule face irrégulière, deux types d'irrégularités apparaissent : celui où la face irrégulière est incidente au sommet central et celui où la face irrégulière est en pointe du patch. En étudiant l'évolution des patches irréguliers (voir [Figure 3.32](#)), il apparaît que le premier cas d'irrégularité génère 5 patches réguliers et 1 patch du second cas d'irrégularité qui lui-même ne génère que des patches réguliers.



**FIGURE 3.32 –** Les deux types de patches irréguliers du schéma *Honeycomb*. L'irrégularité de face centrale devient une irrégularité de pointe et l'irrégularité de pointe disparaît naturellement par la double application des règles de subdivision.

Une fois les patches réguliers et irréguliers définis, il faut trouver la forme du morceau de surface générée par un patch. Les patches du schéma *Honeycomb* étant centrés sur un sommet, chaque patch a trois patches comme voisins directs. Ce qui veut dire que le morceau de surface limite issu de ce patch partage un bord avec trois autres morceaux de surface limite et donc a la forme d'un triangle. Ensuite, un patch génère 6 sous-patches, ayant chacun trois voisins dont deux issus du même patch parent et un issu du patch parent voisin. La paramétrisation qui répond à ces critères est donnée en [Figure 3.33](#). Une fois les différents patches, les transformations, et la paramétrisation du morceau de surface limite définis, il ne reste plus qu'à construire l'automate CIFS correspondant (voir [Figure 3.34](#)).

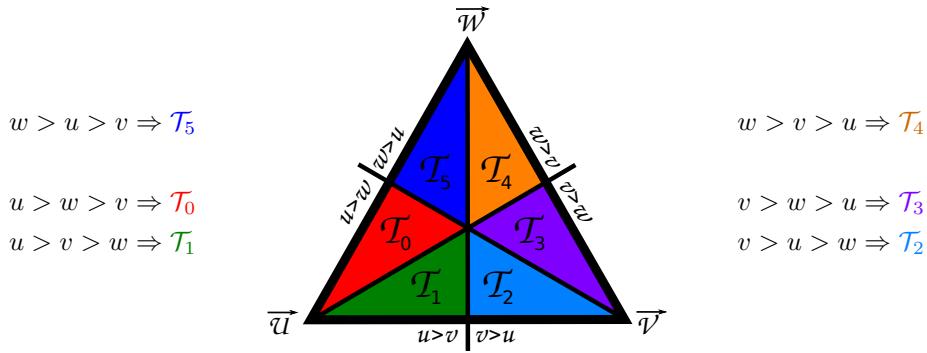


FIGURE 3.33 – Paramétrisation des morceaux de surfaces du schéma *Honeycomb*. L’algorithme de génération des adresses est directement déduit des règles de sélection de la transformation.

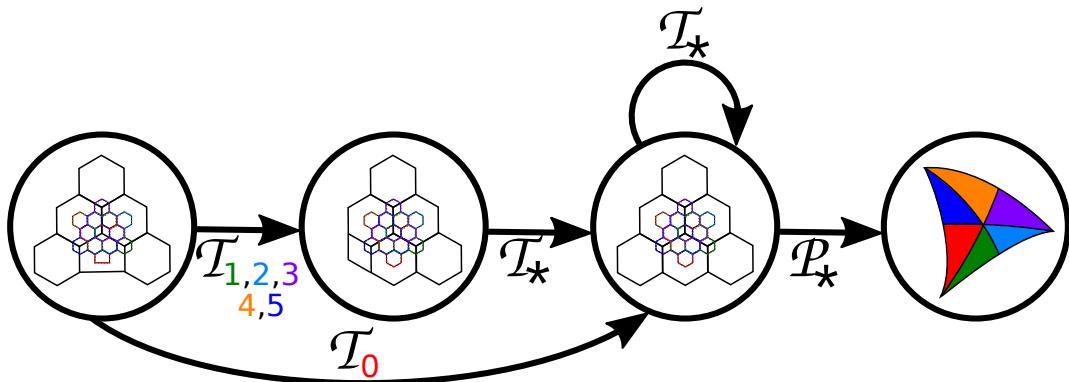


FIGURE 3.34 – Automate CIFS du schéma *Honeycomb*

### 3.8 Le schéma hybride de Catmull-Clark/Loop

En 2003, Jos Stam et Charles T. Loop cherchent à concevoir un schéma primal dont la surface limite est  $C_2$ -continue dans les zones régulières qui puisse être appliqué sur un maillage contenant à la fois des triangles et des quadrangles [SL03]. L’idée principale est d’utiliser les règles du schéma de Catmull-Clark dans les zones composées uniquement de quadrangles et les règles du schéma de Loop dans les zones uniquement triangulaires. Ensuite, il faut trouver les règles de subdivision pour les zones du maillage où des triangles et des quadrangles sont présents (voir Figure 3.35).

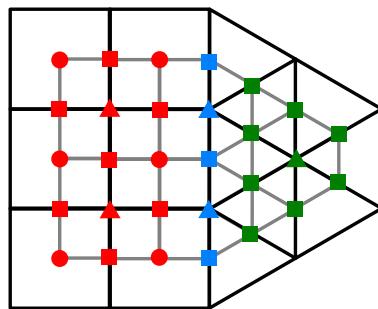


FIGURE 3.35 – Exemple d’une itération du schéma hybride Catmull-Clark/Loop. En rouge les sommets générés par les règles de Catmull-Clark, en vert ceux générés par les règles de Loop, et en cyan ceux dont les règles sont à définir. Les triangles représentent les V-sommets, les carrés les E-sommets et les ronds les F-sommets.

### 3.8.1 Formulation sous forme de masques

En observant le résultat attendu d'une itération du schéma hybride, il apparaît que deux types de masques sont à définir sur la frontière : celui d'une arête entre une face triangulaire et une face quadrangulaire qui génère un E-sommet et celui d'un sommet dont les faces incidentes sont triangulaires et quadrangulaires qui génère un V-Sommet (potentiellement irrégulier)

La base du masque d'arête hybride est le masque d'arête de Catmull-Clark. En effet, les influences des sommets sur l'arête centrale ainsi que celles "côté quadrangle" sont identiques au masque de Catmull-Clark. Ensuite, du "côté triangle", l'influence du sommet en pointe est la somme des influences des deux sommets du quadrangle (les deux sommets ont été fusionnés pour construire un triangle).

Le masque sommet irrégulier est construit de la manière suivante :

- Du "côté quadrangle", les coefficients de Catmull-Clark sont repris
- Du "côté triangle", c'est les coefficients de Loop qui sont repris
- Sur la frontière, l'influence est définie par la moyenne des coefficients de Catmull-Clark et de Loop (38 est la moyenne de 36 et 40, 5 est la moyenne de 6 et 4)

*N.B.* : comme dans la présentation du masque sommet de Loop les coefficients sont exprimés en seizeièmes alors que pour celle de Catmull-Clark ils sont exprimés en soixante-quatrièmes, il faut renormaliser les coefficients de Loop en les multipliant par 4.

La construction du masque pour les sommets irréguliers est décomposée en deux étapes : premièrement un sommet est inséré au milieu de chaque arête et au centre de chaque quadrangle, puis des influences décroissantes sont attribuées au sommet central du patch, aux sommets issus d'une arête, et aux sommets issus d'une face quadrangle. Ces masques sont donnés dans la [Figure 3.36](#).

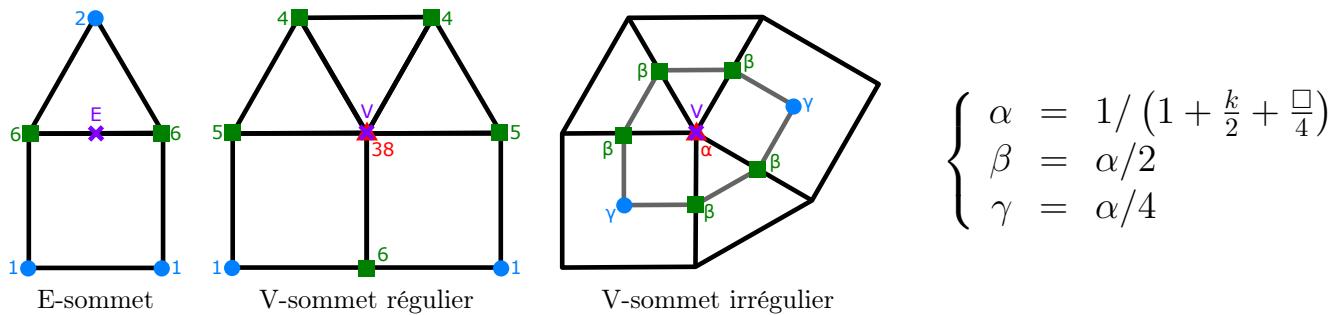


FIGURE 3.36 – Masques du schéma hybride Catmull-Clark/Loop. La valence du sommet central est  $k$  et le nombre de face quadrangles incidentes à ce sommet est  $\square$ .

### 3.8.2 Formulation sous forme d'un automate CIFS

Les patches des schémas de Catmull-Clark et de Loop sont tous les deux définis de la même manière : une face centrale et un anneau de faces qui entoure la face centrale. Dans le cas du schéma hybride, la méthode de construction reste la même.

Dans ce manuscrit, seul le cas où le maillage a une frontière "nette" entre les triangles et les quadrangles (*i.e.* une succession d'arêtes consécutives séparant systématiquement une face triangulaire d'une face quadrangulaire et dont tous les sommets sont de valence 5) est présenté car sinon il y a beaucoup de cas de figure irréguliers. Dans ce cas, la frontière est régulière et l'application des règles de subdivision basiques, éloignent les irrégularités de la frontière au fur et à mesure des itérations. Cela implique que les irrégularités ne sont gérées que par les patches de Catmull-Clark et de Loop, mais pas par les patches hybrides.

Ces patches sont au nombre de trois : celui centré sur une face quadrangulaire, celui centré sur une face triangulaire dont un des sommets appartient à la frontière, et celui centré sur une face triangulaire dont une arête appartient à la frontière. Ces trois patches sont présentés en [Figure 3.37](#).

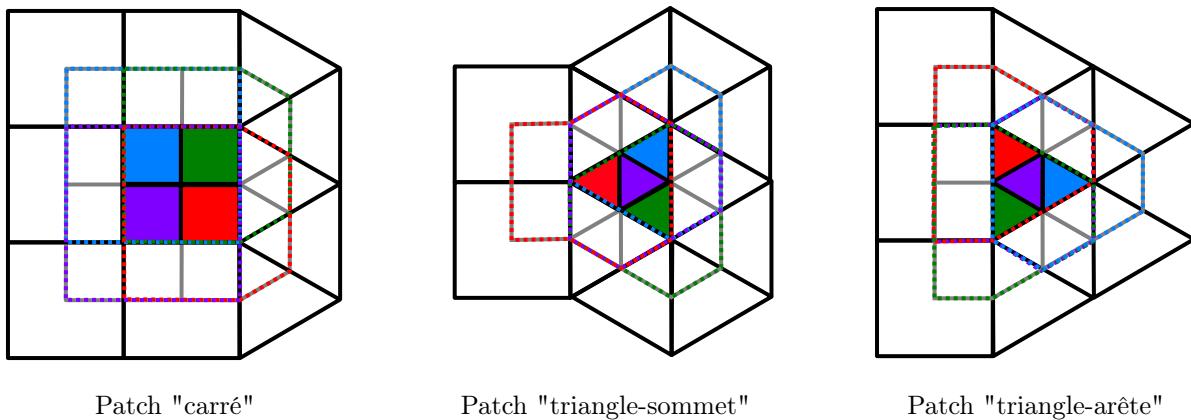


FIGURE 3.37 – Schéma des trois types de patches réguliers du schéma hybride de Catmull-Clark/Loop.

Chaque patch est décomposé en quatre sous-patches par itération :

- le patch "carré" devient deux fois lui-même et deux fois un patch de Catmull-Clark
- le patch "triangle-sommet" devient une fois lui-même et trois fois un patch de Loop
- le patch "triangle-arête" devient deux fois lui-même, une fois un patch "triangle-sommet" et une fois un patch de Loop.

Ces changements de nature entre les patches permet de déduire directement la structure des deux automates CIFS du schéma hybride de Catmull-Clark/Loop (cf. Figure 3.38).

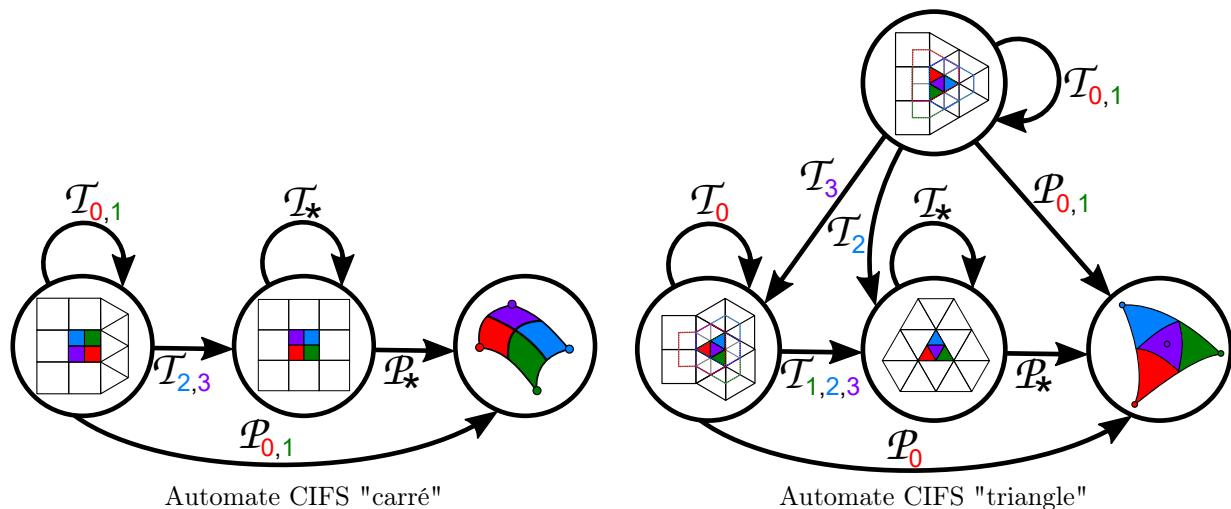


FIGURE 3.38 – Les deux automates CIFS du schéma hybride de Catmull-Clark/Loop

La génération des adresses dépend du patch traité : s'il est centré sur un quadrangle, elle est similaire à celle du schéma de Catmull-Clark (Figure 3.14), s'il est centré sur un triangle c'est celle du schéma de Loop qui est utilisée (Figure 3.28).

### 3.9 Les schémas de subdivision à support non-compact

Les schémas présentés jusqu'à présent sont tous **à support compact** : le patch de contrôle nécessaire et suffisant pour calculer un morceau de surface limite est composé d'un sommet central (pour les schémas duals) ou d'une face centrale (pour les schémas primaux) et d'un anneau de faces qui entoure l'élément central. Ceci implique un nombre limité de positions possibles d'irrégularités à l'intérieur d'un patch.

Dans cette section est présentée la famille des schémas B-Splines uniformes qui sont générés par un algorithme itératif dont le nombre d'itérations dépend du degré de la surface limite. Il est montré que plus ce degré de la surface est élevé, plus le patch de contrôle est étendu et donc plus il y a de cas irréguliers possibles. L'explosion du nombre d'irrégularités entraîne une augmentation du nombre d'états et de transitions de l'automate CIFS ce qui le rend rapidement compliqué à construire manuellement. Cette limitation devrait pouvoir être levée avec une méthode de construction automatique de l'automate CIFS, mais pour l'instant cette méthode n'a pas été définie.

### 3.9.1 L'algorithme de génération des schémas de type B-Splines uniformes

Zorin [ZS01] propose un algorithme général de génération des schémas de type B-Splines uniformes. Ces schémas sont ceux dont la surface limite est une surface tensorielle B-Spline uniforme dans le cas d'un maillage régulier de type grille. Ces schémas sont :

- le schéma bi-linéaire (degré 1)
- le schéma de Doo-Sabin (degré 2) [DS78]
- le schéma de Catmull-Clark bi-quadratique (degré 2) [CC78]
- le schéma de Catmull-Clark classique (degré 3) [CC78]

Il est important de noter qu'il existe 2 schémas de degré 2 : Doo-Sabin qui est le plus courant et une formulation de Catmull-Clark moins répandue. Même si ces deux schémas sont identiques dans le cas d'un maillage régulier (ils produisent tous les deux des surfaces B-Splines bi-quadratiques uniformes), celui utilisé dans [ZS01] est le schéma bi-quadratique de Catmull-Clark.

L'algorithme de génération de Zorin est une généralisation de l'algorithme de Lane-Riesenfeld [LR80] aux maillages de topologie quelconque. L'algorithme de Lane-Riesenfeld génère, pour un degré donné, la méthode de subdivision permettant de construire les B-Splines uniformes de ce degré (cf. page 34). Il est généralisable au cas des surfaces tensorielles B-Splines uniformes en remplaçant les étapes de dédoublement par des étapes de quadruplage (dédoublement selon les deux directions principales) qui créent de nouvelles faces quadrangulaires et les étapes de moyennage d'arête par des étapes de moyennage des anciennes et des nouvelles faces (*i.e.* calcul du centre de gravité). Les différentes étapes de l'algorithme sont présentées en [Figure 3.39](#)

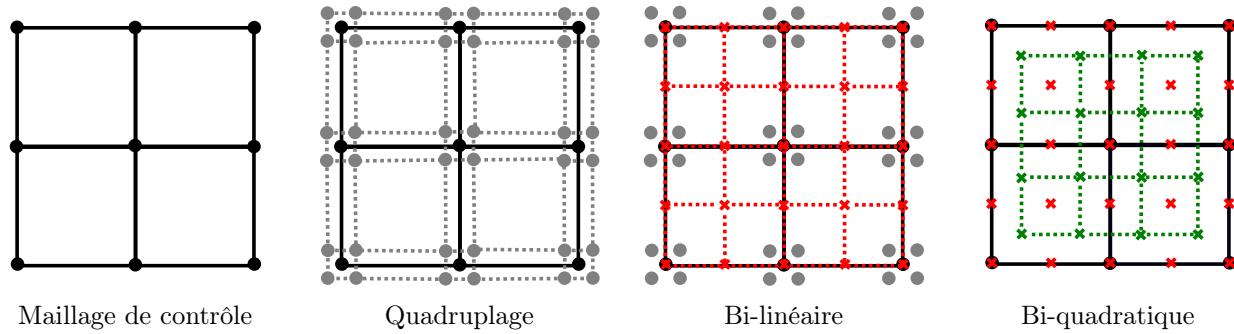


FIGURE 3.39 – Génération de surfaces tensorielles B-Splines uniformes par la méthode du moyennage de faces.

Pour généraliser l'algorithme à des maillages de topologie arbitraire, Zorin remplace simplement l'étape de quadruplage par une étape de multiplication où le sommet est réplié en autant de sommet qu'il a de faces incidentes s'il est à l'intérieur du maillage et en quatre sommets s'il est sur les bords. Les étapes de moyennage restent identiques en plaçant un sommet au centre de gravité de chaque face. Un exemple est donné en [Figure 3.40](#).

### 3.9.2 Les patches des schémas B-Splines uniformes

Une étape de subdivision est constituée d'une étape de multiplication  $\mathbf{D}$  et de  $d$  étapes de moyennage  $\mathbf{M}$  (où  $d$  représente le degré des deux courbes générant la surface par produit tensoriel). La succession des étapes de moyennage impose d'aller chercher des sommets de plus en plus loin, et donc d'avoir des patches de contrôle de plus en plus étendus. Les patches de contrôle pour les degrés 1 à 8 sont donnés en [Figure 3.41](#).

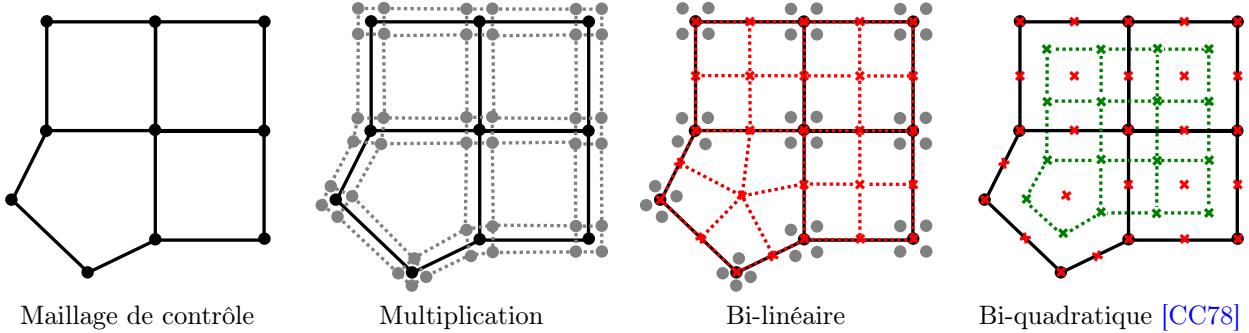


FIGURE 3.40 – Itération d'une surface de subdivision B-Spline quadratique autour d'un patch de Doo-Sabin par l'application d'une multiplication et de deux moyennages de face. Le résultat obtenu est topologiquement identique à la première itération du schéma de Doo-Sabin présenté précédemment.

Plus un patch est étendu, plus il est susceptible de contenir des irrégularités (un sommet de valence différente de 4 ou face non-quadrangle). Même en appliquant suffisamment de fois la subdivision de Zorin pour séparer les irrégularités et imposer qu'aucun patch n'ait plus d'une irrégularité, le nombre des positions possibles d'irrégularités augmente avec le degré. Certaines de ces positions peuvent être regroupées par rotation ou symétrie mais le nombre  $N$  de types de patches irréguliers augmente en suivant la formule suivante :

$$N(d) = \sum_{i=1}^{\lfloor d/2 \rfloor} i$$

Avec l'augmentation du degré de la surface, le nombre de cas irréguliers explose et il devient assez vite compliqué de tester les différents cas à la main et d'en déduire l'automate CIFS et les matrices permettant de gérer les surfaces de haut degrés. Une méthode de génération automatique de l'automate est nécessaire pour pouvoir gérer les surfaces de subdivision de type B-Splines uniformes de degré quelconque.

Au passage, pour la création des schémas de subdivision de type B-Splines interpolants, il est nécessaire de prendre les patches de deux degrés supérieurs pour avoir suffisamment de degré de liberté pour résoudre les contraintes d'interpolation. Par exemple le schéma de Kobbelt [Kob96] qui génère des surfaces interpolantes de degré 3 à partir d'un patch de contrôle, celui de degré 5 avec les trois sommets irréguliers potentiels.

### En résumé :

- Les surfaces de subdivision sont construites à partir d'un maillage de topologie arbitraire sur lequel sont appliquées de manière itérative des règles simples appelées règles de subdivision.
- A notre connaissance, tous les schémas de subdivision sont représentables sous la forme d'automates CIFS barycentriques dont les différents états correspondent aux différentes configurations de patches possibles.
- Mais, dès que les schémas sont de degrés élevés, leur patches ne sont plus à support compact. Ainsi, le nombre de topologie différentes de patches augmente, ce qui entraîne une augmentation conséquente du nombre d'états et de transitions de l'automate et cela le rend difficile à construire. Une méthode de construction automatique de l'automate CIFS pour un degré donné semble la meilleure solution à l'heure actuelle.

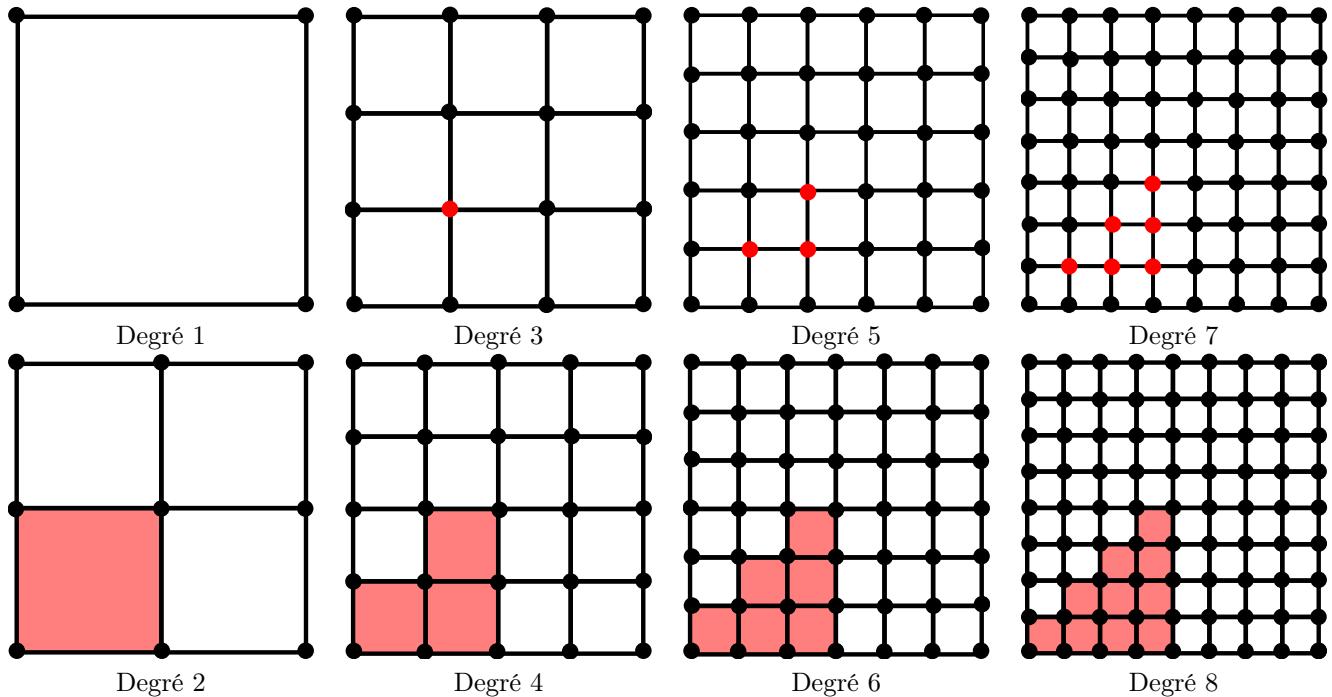


FIGURE 3.41 – Les patches de contrôle des schémas de subdivision de type B-Splines générés par l’algorithme de Zorin. En supposant une unique irrégularité par patch, et à une succession de rotations et de symétries près, les différentes positions possibles de celle-ci sont indiquées en rouge. Pour les schémas impairs, les sommets rouges indiquent un sommet de valence différente de 4. Pour les schémas pairs, la face en rouge est une face potentiellement non-quadrangle.

## Chapitre 4

# B-Splines Rationnelles Non-Uniformes (NURBS)

Dans ce chapitre, il est montré comment représenter les courbes ou les surfaces tensorielles de type NURBS de degré quelconque par un automate CIFS. Cette nouvelle description permet d'intégrer les NURBS, ainsi que tous les modèles CGAO dont elles sont le modèle de représentation, à notre modeleur basé CIFS.

Après avoir expliqué la manière habituelle de calculer une NURBS par la formule de Cox-De-Boor, une manière équivalente, mais moins commune, appelée *blossoming* est présentée. La représentation des NURBS sous forme de *blossoming* nous a permis de développer deux méthodes permettant de représenter les NURBS sous-forme d'un automate CIFS

La première méthode démontre la quasi-auto-similarité des NURBS ce qui permet de déduire l'automate CIFS correspondant. En traitant les cas simples des NURBS quadratiques et cubiques, une méthode de construction des automates CIFS valable quel que soit le degré de la courbe est déduite. Le produit-tensoriel des automates CIFS décrit en [Sous-section 2.4.3](#) est ensuite utilisé pour concevoir les automates CIFS représentant les surfaces tensorielles NURBS de degré quelconque.

La seconde méthode exploite au maximum les possibilités du *blossoming* pour démontrer que chaque morceau d'une NURBS correspond exactement à une B-Spline uniforme. Le *blossoming* de surface tensorielle est ensuite utilisé pour montrer que cette méthode est également valable pour l'uniformisation d'un morceau de surface NURBS tensorielle.

Les résultats décrit dans ce chapitre ont été présentés en 2018 aux communautés française, lors des Journées du Groupe de Travail en Modélisation Géométrique [[MNLG18a](#)], et internationale, lors de la conférence Curves & Surfaces [[MNLG18c](#)]. La première méthode a été publiée dans la conférence *Shape Modeling International* en 2019 [[MGL<sup>+</sup>19](#)] et la seconde fait l'objet d'un article en cours de rédaction [[MGLN19](#)].

## 4.1 Introduction aux NURBS

Les B-Splines Rationnelles Non-Uniformes (NURBS) sont une extension des B-Splines permettant un meilleur contrôle de la courbe obtenue. En effet, certaines formes comme les coniques (*i.e.* intersection d'un cône et d'un plan *e.g.* ellipses, arcs de parabole) ne peuvent être décrites par des B-Splines mais le sont par des NURBS. Grâce à ce contrôle de la forme, elles se sont imposées comme une norme en CGAO et, en tant que telle, elles sont présentes dans la plupart des modeleurs géométriques à but de conception industrielle. C'est pourquoi notre modeleur basé CIFS se doit, lui aussi, de gérer les NURBS.

### 4.1.1 Représentation et calcul des NURBS

Les B-Splines, composées de  $m$  morceaux et de degré  $d$ , vues précédemment sont définies par un polygone de contrôle  $\mathbf{P}$  de  $(d+m)$  points et d'un vecteur nodal  $\mathbf{T}$  composé de  $(2d+m-1)$  noeuds qui agissent sur la forme de la courbe :

$$\begin{cases} \mathbf{P} = [p_0 \dots p_{(d+m-1)}] \\ \mathbf{T} = [t_0 \dots t_{(2d+m-2)}] \end{cases}$$

Les B-Splines étudiées précédemment sont uniformes : l'écart entre chaque noeud du vecteur nodal est constant et de ce fait le vecteur nodal n'est jamais précisé. Dans ce chapitre les B-Splines sont **non-uniformes** (le vecteur nodal agit sur la forme de la courbe) et **rationnelles** (chaque point  $p_i$  du polygone de contrôle possède une masse  $\omega_i$ ). Les B-Splines Non-Uniformes (NUBS) sont calculées par la formule de Cox-De-Boor [Cox72] :

$$\begin{aligned} \mathcal{C}(t) &= \sum_{j=0}^{d+m-1} (N_j^d(t)p_j) \\ \text{avec } N_j^d(t) &= \frac{t - t_j}{t_{(j+d)} - t_j} N_j^{(d-1)}(t) + \frac{t_{(j+d+1)} - t}{t_{(j+d+1)} - t_{(j+1)}} N_{(j+1)}^{(d-1)}(t) \\ N_j^0(t) &= 1 \text{ si } t \in [t_j; t_{j+1}[ \text{ et } 0 \text{ sinon} \end{aligned}$$

Une fonction de base  $N_j^d(t)$  est dite **nulle** si elle est égale à 0 quelque soit la valeur de  $t \in [t_j; t_{j+1}[$  et **active** sinon. En reprenant la formule de Cox-De-Boor, il apparaît qu'une seule fonction de base de degré 0 peut être active à la fois. Chaque fonction de degré  $d$  activant deux fonctions de degré  $(d+1)$  (avec des redondances) il y a toujours  $(d+1)$  fonctions de base de degré  $d$  actives. Ainsi pour une NUBS de degré  $d$ , il y a  $(d+1)$  fonctions de degré  $d$  actives, et donc  $(d+1)$  points de contrôle actifs. Lors du calcul de ces fonctions,  $2d$  noeuds du vecteur nodal interviennent. Le **support** d'un morceau de NUBS correspond à l'ensemble des points de contrôle et des noeuds actifs lors du calcul d'un morceau de courbe (en d'autres termes, l'ensemble des éléments nécessaires et suffisants au calcul de ce morceau de courbe). Lors du passage d'un morceau de NUBS vers le morceau suivant, le premier point de contrôle et le premier noeud sont désactivés et un nouveau point et un nouveau noeud sont activés. L'ensemble des supports des morceaux de NUBS est appelé **support de la NUBS**. Le premier et le dernier noeuds pour lesquels la NUBS est définie sont appelés les **bornes** de la NUBS. Classiquement, les fonctions de base et les coefficients de la formule de Cox-De-Boor sont représentés sous la forme d'un arbre pour une visualisation plus simple de la récurrence du calcul (voir Figure 4.1).

Pour ajouter le concept de **rationalité** aux NUBS (qui deviennent alors les NURBS : B-Splines Rationnelles Non-Uniformes) la formule de Cox-De-Boor est modifiée pour tenir compte des masses des points [Cox72] :

$$\mathcal{C}(t) = \frac{\sum_{j=0}^{d+m-1} (\omega_j N_j^d(t)p_j)}{\sum_{j=0}^{d+m-1} (\omega_j N_j^d(t))}$$

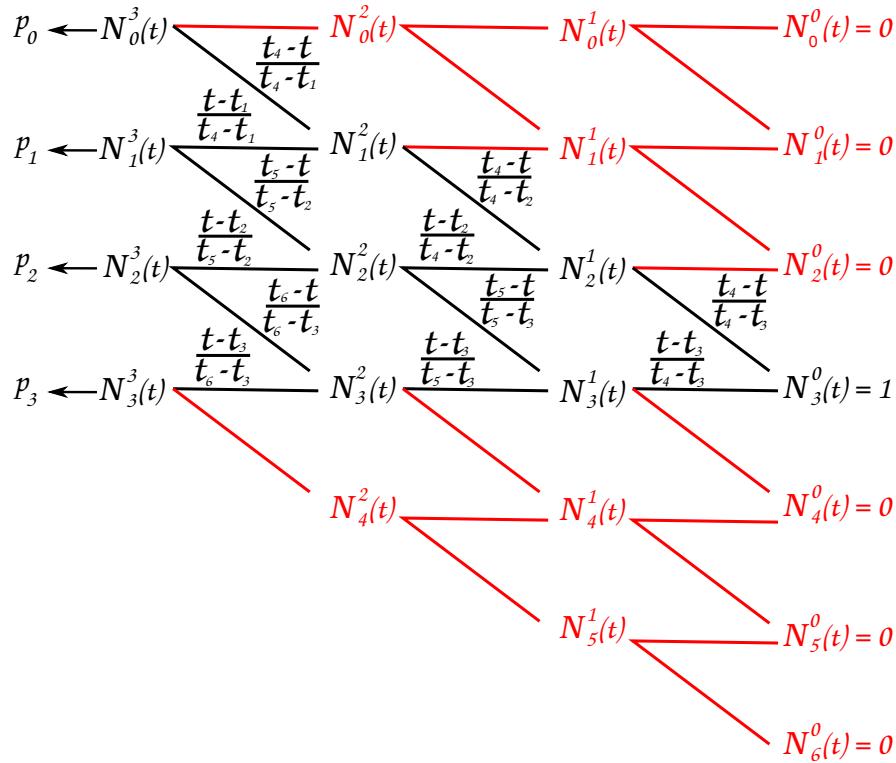


FIGURE 4.1 – Arbre de la construction par récurrence des fonctions de base des B-Splines cubiques non-uniformes. Les fonctions  $N_j^d$  sont en noir lorsqu'elles sont actives et en rouge si elles sont nulles. Dans le cas présent,  $t \in [t_3, t_4[$  et donc la seule fonction de degré 0 active est  $N_3^0$ . Les coefficients associés aux branches noires sont ceux de la formule de récurrence. L'arbre noir qui apparaît à l'inverse de l'arbre global est celui du *blossoming* qui est présenté dans la section suivante.

### Contrôle de la forme : non-uniformité VS rationalité

En plus du polygone de contrôle, deux autres notions agissent sur la forme de la courbe : la **non-uniformité** et la **rationalité**. La non-uniformité, représentée par le **vecteur nodal**, est la plus utilisée des deux notions. Le vecteur nodal permet de gérer finement la paramétrisation de la courbe. Par exemple, l'interpolation de points irrégulièrement espacés avec différentes paramétrisations, donc différents vecteurs noraux, conduit à des courbes aux qualités ou propriétés différentes. La rationalité est obtenue en ajoutant à chaque point du polygone de contrôle une **pondération**. Cette pondération est principalement utilisée pour tracer des coniques car il est impossible de générer une conique avec une B-Spline ou une NURBS non-rationnelle. Deux exemples sont donnés en Figure 4.2.

### Aparté sur les points massiques et pondérations

Les **points massiques** sont une généralisation de la notion de barycentre [FJ98]. L'ensemble de la théorie des points massiques n'est pas nécessaire dans le cas présent mais seulement l'addition qui est une opération commutative et associative.

**Définition.** Soient  $A$  et  $B$  deux points auxquels sont associées les masses respectives  $\omega_A$  et  $\omega_B$ . Les points massiques sont notés  $(A ; \omega_A)$  et  $(B ; \omega_B)$ . La somme de ces deux points massiques est :

$$(A ; \omega_A) + (B ; \omega_B) = \left( \frac{\omega_A}{\omega_A + \omega_B} A + \frac{\omega_B}{\omega_A + \omega_B} B ; (\omega_A + \omega_B) \right)$$

En transformant chaque point du polygone de contrôle  $p_i$  en point massique de masse  $\omega_i$ , et en appliquant la formule des NUBS, le résultat est le même qu'en appliquant la formule de NURBS. En d'autres termes, la formulation des NURBS sur des points de contrôle classique est équivalente à celle des NUBS sur des points massiques. La formulation sous forme de NUBS sur des points massiques est plus simple à mettre en oeuvre car elle sépare les étapes non-uniforme et rationnelle. C'est pourquoi le terme NUBS sera privilégié dans la suite du

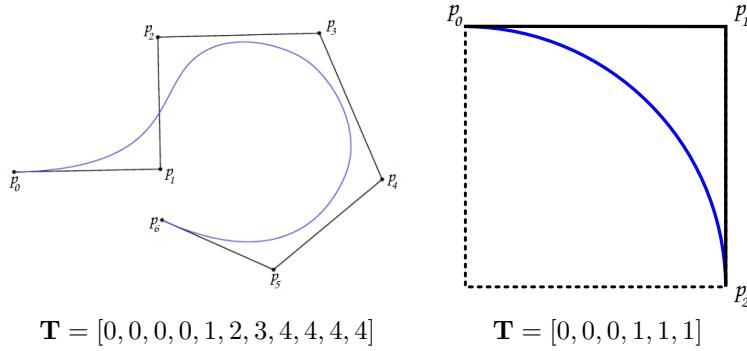


FIGURE 4.2 – Comparaison entre la non-uniformité et la rationalité. A gauche une courbe NUBS et à droite un quart de cercle tracé par une NURBS dont les pondérations sont  $w_0 = w_2 = \sqrt{2}$  et  $w_1 = 1$ .

manuscrit pour éviter toute confusion mais il faut bien garder à l'esprit que si des points massiques sont utilisés, la courbe définie est une NURBS.

### Vecteur inter-nodal

De plus, il faut préciser que dans la formule de Cox-De-Boor, ce n'est pas les valeurs des noeuds qui influent sur la forme de la courbe mais les ratios entre les différences de deux noeuds consécutifs. Ainsi, ajouter ou multiplier un même scalaire à toutes les valeurs du vecteur ne change pas la courbe définie. C'est pourquoi, le vecteur nodal  $\mathbf{T}$  est souvent remplacé par un vecteur inter-nodal  $\mathbf{U} = [u_0 \dots u_{2d+m-3}]$  dont chaque valeur inter-nodale est la différence de deux valeurs nodales consécutives :  $u_i = t_{(i+1)} - t_i$ . Une NURBS peut donc aussi être définie par le couple :

$$\begin{cases} \mathbf{P} &= [p_0 \dots p_{(d+m-1)}] \\ \mathbf{U} &= [u_0 \dots u_{(2d+m-3)}] \end{cases}$$

Les représentations  $(\mathbf{P}, \mathbf{T})$  et  $(\mathbf{P}, \mathbf{U})$  étant équivalentes, la plus adaptée sera utilisée selon la situation. Parfois elles seront même utilisées en parallèle pour offrir deux approches sur le même point.

### 4.1.2 Blossoming (floraison)

Les NUBS sont présentées ci-dessus par la représentation habituelle qu'est la formule de Cox-De-Boor [Cox72] mais dans ce manuscrit la formulation *blossoming* (floraison) de Lyle Ramshaw [Ram87] est privilégiée car cette représentation fait apparaître naturellement l'auto-similarité des NUBS. Le *blossoming* est également valable pour des Splines qui ne sont ni des B-Splines uniformes, ni des NUBS mais ce ne sera pas le cas dans ce manuscrit.

**Définition.** La floraison d'une courbe de degré  $d$  est représentée par une **étiquette** de  $d$  arguments :  $\{t_i \dots t_{i+d-1}\}$ . Le blossoming est défini par trois propriétés de base : **symétrie**, **diagonalité**, et **multi-affinité** auxquelles est ajoutée une nouvelle déduite des trois précédentes : la **consécutivité**.

**Symétrie :**

L'ordre des arguments d'une étiquette n'a aucune incidence sur le blossoming auquel elle correspond. Ceci permet de simplifier la lecture en écrivant systématiquement les arguments en ordre croissant :

$$\{\dots t_i \dots t_j \dots\} = \{\dots t_j \dots t_i \dots\} \quad (j < i)$$

**Diagonalité :**

Tout point de la courbe limite (dans notre cas de la NUBS) correspond à une étiquette dont tous les arguments ont la même valeur. Cette valeur est le paramètre de la courbe qui correspond à ce point :

$$\mathcal{C}(t) = \{t \dots t\}$$

### Multi-affinité :

Chaque étiquette peut être définie comme une combinaison affine de deux autres étiquettes si ces trois étiquettes ont tous leur arguments, sauf un, identiques. Les poids de la combinaison affine des étiquettes correspondent aux poids de la combinaison affine de ces arguments qui diffèrent :

$$\{\dots t \dots\} = \frac{b-t}{b-a}\{\dots a \dots\} + \frac{t-a}{b-a}\{\dots b \dots\}$$

### Consécutivité :

Une étiquette dont les valeurs des arguments correspondent aux valeurs de noeuds consécutifs du vecteur nodal correspond au point du polygone de contrôle dont l'indice est le même que celui du premier argument :

$$p_i : \{t_i, t_{i+1} \dots t_{i+d-1}\}$$

Tout point de la courbe  $C(t)$  peut être exprimé comme une combinaison barycentrique des points du polygone de contrôle  $\mathbf{P}$  en appliquant directement les propriétés du *blossoming* :

- Premièrement la propriété de diagonalité est utilisée pour construire l'étiquette de  $C(t) : \{t \dots t\}$
- Ensuite la propriété de multi-affinité est appliquée sur l'étiquette-mère pour la "découper" en deux étiquettes-filles. Pour cela, un des arguments  $t$  de la mère est découpé en deux nouveaux arguments  $t_i$  qui est le plus grand noeud du vecteur nodal inférieur à tous les arguments de l'étiquette-mère et  $t_j$  qui est le plus petit noeud supérieur à tous les arguments de l'étiquette-mère. La propriété de multi-affinité permet d'exprimer l'étiquette-mère en fonction des étiquettes-filles.
- L'étape précédente est répétée sur chaque étiquette qui possède encore au moins un argument  $t$ .
- Lorsque l'algorithme se termine, il ne reste plus que des étiquettes respectant la propriété de consécutivité et donc des points de contrôle.

Cet algorithme est le plus souvent représenté sous la forme d'un arbre (d'où le nom de floraison) où la **racine** est l'étiquette du point de la courbe, les **noeuds** des étiquettes intermédiaires, et les **feuilles** les étiquettes des points de contrôle (voir Figure 4.3). Un **coefficients** est associé à chaque **branche** qui relie une étiquette-mère à une étiquette-fille : ce coefficient est l'influence de la fille sur la mère. L'influence d'un **chemin**, qui relie la racine à l'une des feuilles, sur le point de la courbe calculé est le produit des coefficients associés aux branches traversées par le chemin. L'influence d'un point de contrôle sur le point de la courbe est la somme de l'influence des chemins qui se terminent sur la feuille correspondante.

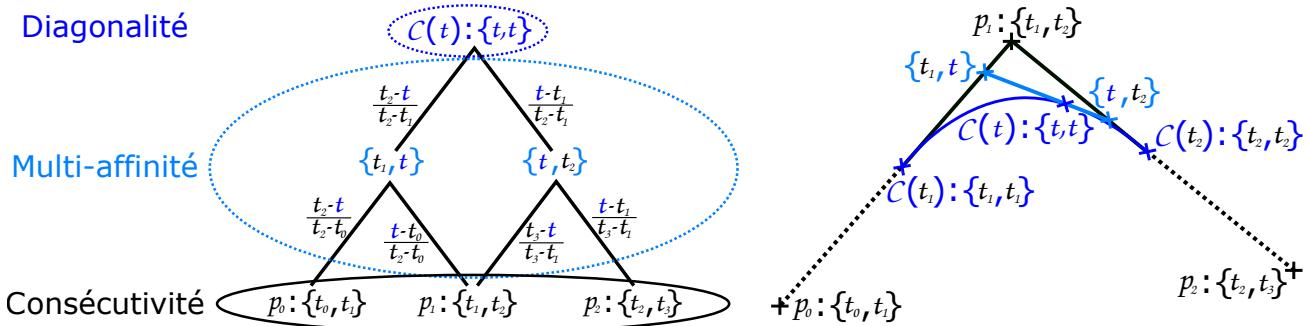


FIGURE 4.3 – Exemple de *blossoming* sur une B-Spline quadratique non-uniforme. Le point  $C(t)$ ,  $t \in [t_1; t_2]$  est exprimé en fonction des points de contrôle  $p_0$ ,  $p_1$ , et  $p_2$ . À gauche l'arbre *blossoming* permettant le calcul des coefficients et à droite la représentation dans le domaine géométrique. Sur la figure de droite les différents points intervenant dans la construction de  $C(t)$  sont représentés par des croix : en noir les points de contrôle de départ, en cyan les points intermédiaires, et en bleu le point  $C(t)$ . Les segments noirs pleins représentent les deux arêtes sur lesquelles les croix cyans se déplacent lorsque  $t$  varie entre  $t_1$  et  $t_2$ . La courbe bleue est la B-Spline  $C(t)$  obtenue en faisant varier  $t \in [t_1; t_2]$  et les croix bleues représentent les extrémités de la courbe ( $t = t_1$  et  $t = t_2$ ).

La construction géométrique (Figure 4.3 droite) permettant le calcul d'un point de la NUBS est similaire au méthode de De Casteljau (voir Figure 2.11 page 32) et de Lane-Riesenfeld (page 35). Il est donc naturel de penser à utiliser un algorithme similaire pour afficher une NUBS. Cet algorithme, appelé **Lane-Riesenfeld généralisé** [CHR13], est basé sur l'insertion de nouveaux noeuds dans le vecteur nodal.

### 4.1.3 Insertion d'un noeud

Une NUBS de degré  $d$  et de  $m$  morceaux est définie par un polygone de contrôle  $\mathbf{P}$  de  $(d + m)$  points et un vecteur nodal  $\mathbf{T}$  de  $(2d + m - 1)$  noeuds. L'objectif est de modifier le vecteur nodal sans modifier la NUBS et pour cela c'est le polygone de contrôle qui doit s'adapter. Dans le cas présent un nouveau noeud est ajouté dans le vecteur nodal ( $\mathbf{T}$  devient  $\mathbf{T}'$ ) et il faut trouver le nouveau polygone de contrôle  $\mathbf{P}'$  qui associé à  $\mathbf{T}'$  définit la même NUBS que la paire  $(\mathbf{P}, \mathbf{T})$ . Ce nouveau polygone est trouvé par *blossoming* : chaque point de  $\mathbf{P}'$  est défini comme une étiquette dont les arguments sont des noeuds consécutifs de  $\mathbf{T}'$ , puis cette étiquette est décomposée en étiquettes dont les arguments sont des noeuds consécutifs de  $\mathbf{T}$ . Ainsi tous les points de  $\mathbf{P}'$  sont définis par une combinaison barycentrique des points de  $\mathbf{P}$ .

Selon la zone où le nouveau noeud est inséré, le comportement de la courbe n'est pas le même :

- si le noeud est inséré entre les bornes de la NUBS, celle-ci est alors composée d'un morceau supplémentaire
- si le noeud est inséré en dehors des bornes, la NUBS est redéfinie mais conserve son nombre de morceaux.

Un exemple d'insertion de noeud dans une NUBS quadratique à un morceau défini par le couple  $(\mathbf{P} = [p_0, p_1, p_2], \mathbf{T} = [t_0, t_1, t_2, t_3])$  est donné pour chacun des deux cas dans les sous-sections suivantes ; ainsi qu'une explication du cas général.

#### Cas numéro 1 : insertion d'un noeud entre les bornes

La NUBS est définie dans l'intervalle nodal  $[t_1; t_2]$  et donc un nouveau noeud  $t_{1b}$  est ajouté dans cet intervalle, ainsi  $\mathbf{T}$  devient  $\mathbf{T}' = [t_0, t_1, t_{1b}, t_2, t_3]$ . Les étiquettes des points de  $\mathbf{P}'$  sont  $\{t_0, t_1\}$ ,  $\{t_1, t_{1b}\}$ ,  $\{t_{1b}, t_2\}$ ,  $\{t_2, t_3\}$ . Le premier et le dernier point sont déjà définis : il s'agit de  $p_0$  et  $p_2$ . Les deux autres points sont exprimés en fonction des points de  $\mathbf{P}$  (les coefficients sont issus de la Figure 4.3) :

$$p'_1 : \{t_1, t_{1b}\} = \frac{t_2 - t_{1b}}{t_2 - t_0} p_0 + \frac{t_{1b} - t_0}{t_2 - t_0} p_1 \quad ; \quad p'_{1b} : \{t_{1b}, t_2\} = \frac{t_3 - t_{1b}}{t_3 - t_1} p_1 + \frac{t_{1b} - t_1}{t_3 - t_1} p_2$$

Le polygone de contrôle et le vecteur nodal comportent désormais un élément de plus, la courbe possède donc un morceau supplémentaire. L'exemple de la NUBS quadratique est donné en Figure 4.4.

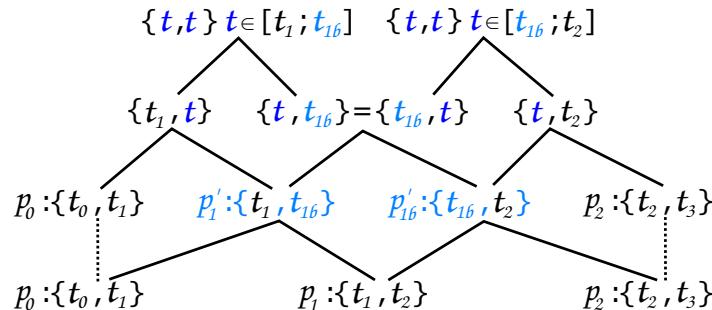


FIGURE 4.4 – *Blossoming* de l'insertion d'un noeud entre les bornes d'une NUBS quadratique. Le nouveau polygone de contrôle (deuxième ligne en partant du bas) est défini en fonction de l'ancien (ligne du bas). La NUBS (définie sur  $[t_1; t_2]$ ) qui avait pour support  $[p_0, p_1, p_2]$  a été découpée en deux morceaux ( $t \leq t_{1b}$  et  $t \geq t_{1b}$ ) qui ont respectivement pour support  $[p_0, p'_1, p'_{1b}]$  et  $[p'_1, p'_{1b}, p_2]$ .

D'une manière plus générale : lorsqu'un noeud est ajouté dans un intervalle où la courbe est déjà définie, la courbe est augmentée d'un point de contrôle, d'un noeud et par conséquent d'un morceau de courbe. En effet, découper en deux un intervalle nodal où la courbe est définie entraîne la création de deux nouveaux intervalles où la courbe est également définie. Il y a donc un intervalle supplémentaire et par conséquent un morceau de courbe en plus.

### Cas numéro 2 : insertion d'un noeud en dehors des bornes

Un nouveau noeud  $t$  est ajouté à  $\mathbf{T}$  tel que  $\mathbf{T}' = [t_0, t_{0b}, t_1, t_2, t_3]$ . Les étiquettes des points de  $\mathbf{P}'$  sont  $\{t_0, t_{0b}\}$ ,  $\{t_{0b}, t_1\}$ ,  $\{t_1, t_2\}$ ,  $\{t_2, t_3\}$ . De manière similaire au cas numéro 1, les deux derniers points sont  $p_1$  et  $p_2$  et le deuxième point est calculé :

$$p'_{0b} : \{t_{0b}, t_1\} = \frac{t_2 - t_{0b}}{t_2 - t_0} p_0 + \frac{t_{0b} - t_0}{t_2 - t_0} p_1$$

Lors du calcul de l'étiquette  $\{t_0, t_{0b}\}$  il apparaît qu'elle est inutile. En effet, en construisant le *blossoming*, cette étiquette sert de support au morceau de courbe  $\mathcal{C}(t), t \in [t_{0b}; t_1]$  qui n'existe pas. Elle est donc supprimée ainsi que toutes les branches qui conduisent à elle. Donc le nouveau couple qui définit la courbe est ( $\mathbf{P}' = [p'_{0b}, p_1, p_2]$ ,  $\mathbf{T}' = [t_{0b}, t_1, t_2, t_3]$ ) (voir Figure 4.5).

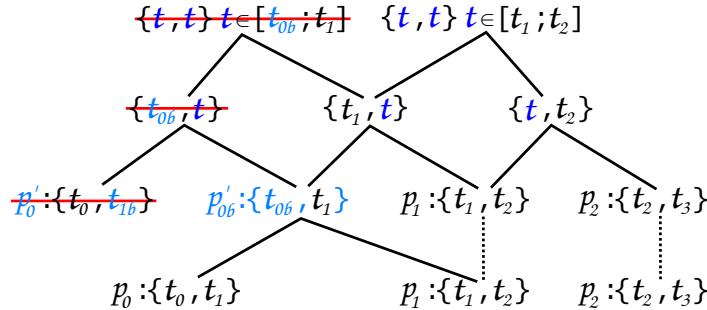


FIGURE 4.5 – *Blossoming* de l'insertion d'un nouveau noeud dans un intervalle nodal où la courbe n'est pas définie. En construisant l'arbre, il apparaît que le point  $p'_0$  ne sert pas de support à la courbe ( $\mathcal{C}(t), t < t_1$ ). Les branches pendantes sont élaguées (barrées en rouge dans la figure).

D'une manière plus générale, lorsqu'un noeud est ajouté à gauche des intervalles où la courbe est définie, le point le plus à gauche du nouveau polygone de contrôle correspond à une étiquette inutile et est donc supprimé. Ceci implique que le noeud le plus à gauche est également supprimé. Le raisonnement est évidemment identique à droite. Cela a pour conséquence que la courbe conserve son nombre de morceaux.

#### En résumé :

Le vecteur inter-nodal d'une NUBS de degré  $d$  et de  $m$  morceaux est composé de  $(d - 1)$  intervalles à gauche, de  $m$  intervalles où la NUBS est définie (un intervalle par morceau) et de  $(d - 1)$  intervalles à droite. Les bornes de la courbe sont fixes : insérer des noeuds dans cette zone n'entraîne qu'un découpage en deux du morceau de NUBS correspondant à l'intervalle d'insertion (cas numéro 1). Le nombre d'intervalles à gauche et à droite de la zone de définition est fixe : l'insertion d'un noeud dans un de ces intervalles crée un intervalle "en trop" ce qui entraîne la suppression du noeud/l'intervalle extrême de ce côté (cas numéro 2). Les deux cas sont présentés dans la Figure 4.6.

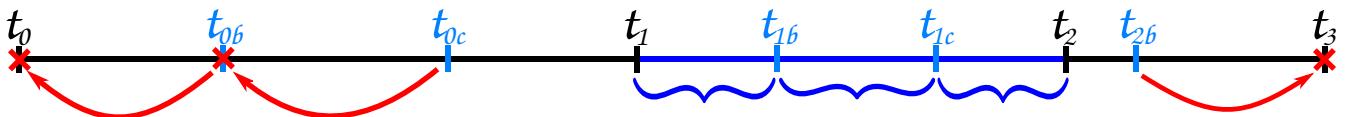


FIGURE 4.6 – Différents exemples des conséquences d'une insertion de noeuds dans un vecteur nodal. De nouveaux noeuds sont insérés dans le vecteur nodal quadratique  $[t_0, t_1, t_2, t_3]$ . Les nouveaux noeuds  $t_{0b}$ ,  $t_{0c}$ , et  $t_{2b}$  entraînent respectivement la suppression (en rouge sur le schéma) des noeuds  $t_0$ ,  $t_{0b}$ , et  $t_3$ . Les nouveaux noeuds  $t_{1b}$  et  $t_{1c}$  découpent la zone de définition de la NUBS ( $t \in [t_1; t_2]$ ) en trois zones (représentées par les accolades bleues sur le schéma). Le nouveau vecteur nodal est  $[t_{0c}, t_1, t_{1b}, t_{1c}, t_2, t_{2b}]$ .

### Stratégies d'insertion de noeuds :

Maintenant que les insertions de noeuds et leurs conséquences ont été définies, il reste encore à définir qu'elle est la meilleure stratégie pour insérer ces noeuds. L'objectif ultime lors de ces insertions serait d'obtenir un vecteur nodal uniforme. Chacune des deux sections suivantes présente une stratégie différente et les CIFS correspondant :

- Stratégie numéro 1 : l'insertion d'un noeud au milieu de chaque intervalle nodal ;
- Stratégie numéro 2 : l'uniformisation préalable du vecteur nodal.

Les avantages et limitations de chacune des méthodes sont présentés dans la section correspondante.

## 4.2 Stratégie numéro 1 : insertion au milieu de chaque intervalle nodal

Cette stratégie est la première étudiée dans le cadre de cette thèse. Les résultats obtenus ont été publiés à la conférence *Shape Modeling International* qui eu lieu à Vancouver en 2019 [MGL<sup>+</sup>19].

Comme son nom l'indique, cette stratégie consiste à insérer un nouveau noeud strictement au milieu de chaque intervalle nodal, le transformant ainsi en deux intervalles noraux consécutifs de même longueur. En répétant l'opération les **zones d'uniformité** (séquence de noeuds consécutifs séparés par des intervalles de longueur constante) sont composées de plus en plus de noeuds jusqu'à permettre la construction de morceaux de courbe uniformes (lorsque  $2d$  noeuds consécutifs sont espacés de manière uniforme).

L'insertion au milieu de chaque intervalle nodal est le point de vue classique sur cette stratégie mais un autre point de vue équivalent est possible : celui du **dédoublement des inter-noeuds**.

**Démonstration.** Soit  $\mathbf{T} = [\dots a, b, c, d \dots]$  une sous-partie d'un vecteur nodal et  $\mathbf{U} = [\dots b - a, c - b, d - c \dots]$  la sous-partie du vecteur inter-nodal correspondant

$$\begin{aligned} \mathbf{T} = [\dots a, b, c, d \dots] &\rightarrow \mathbf{T}' = [\dots a, \frac{a+b}{2}, b, \frac{b+c}{2}, c, \frac{c+d}{2}, d \dots] \quad // \text{insertion au milieu} \\ &\rightarrow \mathbf{T}' = [\dots 2a, a+b, 2b, b+c, 2c, c+d, 2d \dots] \quad // \text{multiplication par 2} \\ \mathbf{U} = [\dots b-a, c-b, d-c \dots] &\rightarrow \mathbf{U}' = [\dots b-a, b-a, c-b, c-b, d-c, d-c \dots] \end{aligned}$$

Comme ce qui compte est le rapport entre les intervalles noraux consécutifs et non la longueur de ceux-ci, multiplier toutes les valeurs par deux n'a aucun influence sur la courbe mais permet de conserver les valeurs existant avant l'insertion au milieu. Et donc il apparaît que l'insertion au milieu de chaque intervalle nodal correspond à un dédoublement de l'inter-noeud correspondant. Les zones d'uniformité sont alors les séquences d'inter-noeuds consécutifs de même valeur.

□

Il faut également prendre en compte le fait que les noeuds insérés ne le sont pas toujours entre les bornes de la courbe. En effet, le vecteur inter-nodal d'une NURBS de degré  $d$  est composé de  $(2d + m - 2)$  inter-noeuds. En réorganisant le nombre d'inter-noeuds, il est composé de  $(d - 1)$  inter-noeuds à gauche, puis de  $m$  inter-noeuds au centre (là où la NURBS est définie), et enfin de  $(d - 1)$  inter-noeuds à droite. Il y a alors deux cas de figure distincts :

- Chacun des  $m$  inter-noeuds dédoublés au centre coupe en deux un morceau de courbe et donc le nombre de morceaux de la NURBS est doublé.
- A l'inverse, les  $(d - 1)$  inter-noeuds à gauche et à droite se dédoublent en insérant des noeuds en dehors des bornes, ce qui a pour conséquence la suppression des  $(d - 1)$  inter-noeuds les plus à gauche et à droite après le dédoublement. En répétant l'opération suffisamment de fois, les  $(d - 1)$  inter-noeuds les plus à gauche (respectivement à droite) deviennent identiques et sont conservés d'un dédoublement à l'autre.

### Exemple :

Soit une NURBS de degré 4 composée de 3 morceaux dont le vecteur inter-nodal est  $[a, b, c, i, j, k, x, y, z]$ . Ce vecteur évolue de la manière suivante :

Vecteur de départ :	$[a, b, c, i, j, k, x, y, z]$
1er dédoublement :	$[b, c, c, i, i, j, k, k, x, x, y]$
2ème dédoublement :	$[c, c, c, i, i, i, i, j, j, j, k, k, k, x, x, x]$
3ème dédoublement :	$[c, c, c, i, i, i, i, i, i, j, j, j, j, k, k, k, k, k, k, x, x, x]$

Après suffisamment de dédoublement, le vecteur inter-nodal n'est plus composé que de  $(2+m)$  valeurs différentes et son comportement devient stationnaire (bien que le nombre de morceaux de courbe continue de doubler). Comme vu jusqu'à présent, un comportement stationnaire peut être représenté par un CIFS.

### 4.2.1 CIFS de dédoublement d'inter-noeuds

Comme chaque morceau d'une NUBS peut être traité indépendamment des autres, le CIFS sera construit uniquement pour les NUBS d'un seul morceau. Pour rappel, celle-ci est décrite par le couple :

$$\begin{cases} \mathbf{P} = [p_0 \dots p_d] \\ \mathbf{U} = [u_0 \dots u_{(2d-2)}] \end{cases}$$

Le dédoublement des inter-noeuds transforme une courbe d'un seul morceau en une courbe de deux morceaux de même degré ou en deux courbes d'un seul morceau de ce degré. Dans le second cas, il y a auto-similarité. Le CIFS est donc composé des deux transformations  $\mathcal{L}$  et  $\mathcal{R}$  qui associent une courbe à une autre courbe. Par conséquent, ces transformations agissent à la fois sur le vecteur inter-nodal et sur le polygone de contrôle :

$$\begin{aligned} \mathcal{L} : (\mathbf{P}; \mathbf{U}) &\mapsto (\mathbf{P}_{\mathcal{L}}; \mathbf{U}_{\mathcal{L}}) \\ \mathcal{R} : (\mathbf{P}; \mathbf{U}) &\mapsto (\mathbf{P}_{\mathcal{R}}; \mathbf{U}_{\mathcal{R}}) \end{aligned}$$

### 4.2.2 Le vecteur inter-nodal

Lors du dédoublement, la taille du vecteur inter-nodal passe de  $(2d - 1)$  à  $2d$ . Les transformations  $\mathcal{L}$  et  $\mathcal{R}$  extraient respectivement les  $(2d - 1)$  inter-noeuds les plus à gauche et les plus à droite (avec  $(2d - 2)$  inter-noeuds en commun). Un exemple de l'application de ces transformations sur un vecteur inter-nodal quadratique est donné en Figure 4.7.

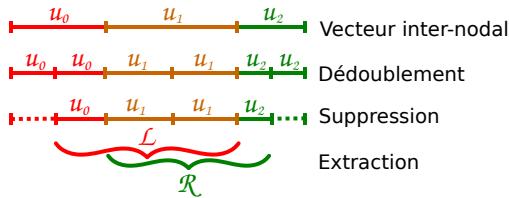


FIGURE 4.7 – Schéma des transformations  $\mathcal{L}$  et  $\mathcal{R}$  sur un vecteur inter-nodal quadratique.

### 4.2.3 Les polygones de contrôle

Chaque transformation associe le vecteur inter-nodal à un nouveau vecteur. Ce nouveau vecteur correspond à de nouvelles étiquettes consécutives et donc à un nouveau polygone de contrôle. Ces étiquettes sont calculées en fonction des anciennes par *blossoming* (voir Figure 4.8).

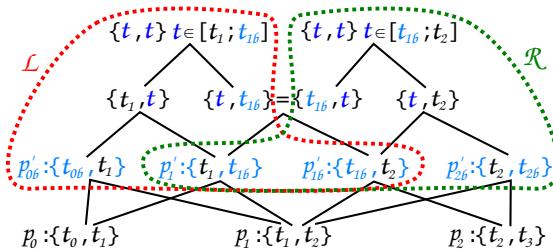


FIGURE 4.8 – *Blossoming* permettant l'expression d'un nouveau polygone de contrôle quadratique en fonction de l'ancien. La ligne du bas représente l'ancien polygone de contrôle et celle juste au-dessus le nouveau. Le nouveau noeud  $t_{1b}$  ayant été inséré dans l'ancien intervalle  $[t_1, t_2]$ , la courbe est désormais composé de deux morceaux :  $t \in [t_1, t_{1b}]$  et  $t \in [t_{1b}, t_2]$ . Chacun de ses deux morceaux repose sur trois points du nouveau polygone de contrôle (sur la figure le premier morceau est entouré en rouge et le second en vert).

Comme des noeuds ont été insérés dans chaque intervalle il existe deux types d'étiquettes consécutives : celles qui commencent par un ancien noeud et celles qui commencent par un nouveau. Il faut donc exprimer ces deux types d'étiquettes en fonction des anciennes étiquettes consécutives (*i.e.* les anciens points de contrôle). Les matrices sont ensuite construites avec les coefficients trouvés par *blossoming*. Les *blossomings* et les matrices de subdivision correspondantes des cas quadratiques et cubiques sont présentés dans les Figures 4.9, 4.10, et 4.11.

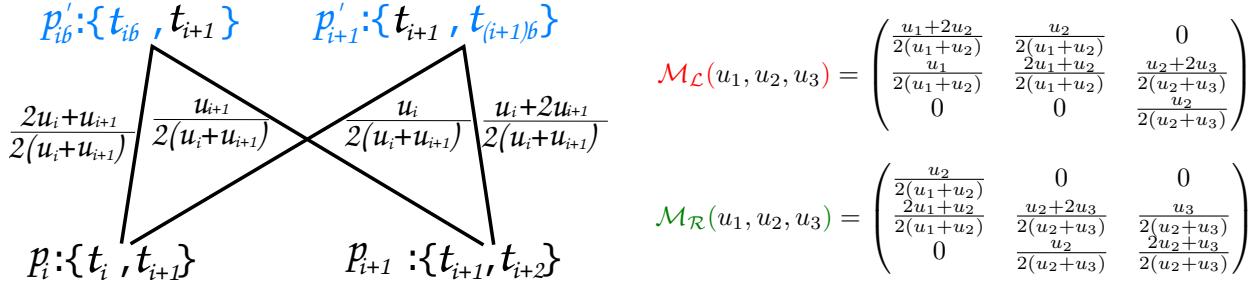


FIGURE 4.9 – Calcul des nouveaux polygones de contrôle en fonction des anciens pour les NUBS quadratiques. A gauche le calcul par *blossoming* et à droite les matrices  $\mathcal{M}_L$  et  $\mathcal{M}_R$  déduites.

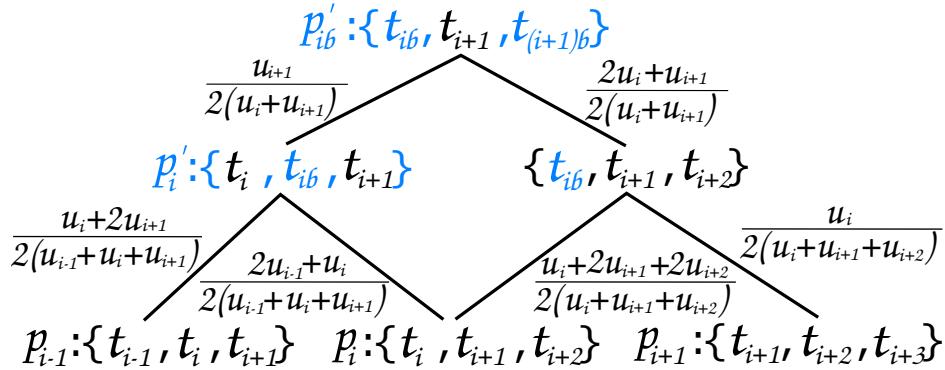


FIGURE 4.10 – *Blossoming* des nouveaux points de contrôle en fonction des anciens pour les NUBS cubiques.

$$\mathcal{M}_L(u_1, \dots, u_4) = \begin{pmatrix} \frac{u_1+2u_2}{2(u_0+u_1+u_2)} & \frac{2u_0+u_1}{2(u_0+u_1+u_2)} & 0 & 0 \\ \frac{u_2}{2(u_1+u_2)} \frac{u_1+2u_2}{2(u_0+u_1+u_2)} & \frac{u_2}{2(u_1+u_2)} \frac{2u_0+u_1}{2(u_0+u_1+u_2)} + \frac{2u_1+u_2}{2(u_1+u_2)} \frac{u_1+2u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{2u_1+u_2}{2(u_1+u_2)} \frac{u_1}{2(u_1+u_2+u_3)} & 0 \\ 0 & \frac{u_2+2u_3}{2(u_2+u_3)} \frac{u_1+2u_2+2u_3}{2(u_1+u_2+u_3)} & 0 & 0 \\ 0 & \frac{u_3}{2(u_2+u_3)} \frac{u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{u_3}{2(u_2+u_3)} \frac{2u_1+u_2}{2(u_1+u_2+u_3)} + \frac{2u_2+u_3}{2(u_2+u_3)} \frac{u_2+2u_3+2u_4}{2(u_2+u_3+u_4)} & \frac{2u_2+u_3}{2(u_2+u_3)} \frac{u_2}{2(u_2+u_3+u_4)} \end{pmatrix}$$

$$\mathcal{M}_R(u_1, \dots, u_4) = \begin{pmatrix} \frac{u_2}{2(u_1+u_2)} \frac{u_1+2u_2}{2(u_0+u_1+u_2)} & \frac{2u_0+u_1}{2(u_0+u_1+u_2)} + \frac{2u_1+u_2}{2(u_1+u_2)} \frac{u_1+2u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{2u_1+u_2}{2(u_1+u_2)} \frac{u_1}{2(u_1+u_2+u_3)} & 0 \\ 0 & \frac{u_2+2u_3}{2(u_2+u_3)} \frac{u_1+2u_2+2u_3}{2(u_1+u_2+u_3)} & 0 & 0 \\ 0 & \frac{u_3}{2(u_2+u_3)} \frac{u_2+2u_3}{2(u_1+u_2+u_3)} & \frac{u_3}{2(u_2+u_3)} \frac{2u_1+u_2}{2(u_1+u_2+u_3)} + \frac{2u_2+u_3}{2(u_2+u_3)} \frac{u_2+2u_3+2u_4}{2(u_2+u_3+u_4)} & \frac{2u_2+u_3}{2(u_2+u_3)} \frac{u_2}{2(u_2+u_3+u_4)} \\ 0 & 0 & \frac{u_3+2u_4}{2(u_2+u_3+u_4)} & \frac{2u_2+u_3}{2(u_2+u_3+u_4)} \frac{u_2}{2(u_2+u_3+u_4)} \end{pmatrix}$$

FIGURE 4.11 – Matrices de l'automate CIFS de NUBS cubique.

#### 4.2.4 Les automates CIFS de NUBS

Les transformations du CIFS transforment à la fois le polygone de contrôle et le vecteur inter-nodal. Le nouveau vecteur inter-nodal dépend **uniquement** de l'ancien vecteur inter-nodal alors que le nouveau polygone dépend **à la fois** de l'ancien polygone de contrôle **ET** de l'ancien vecteur inter-nodal. Il faut donc d'abord étudier l'évolution du vecteur inter-nodal puis celle du polygone de contrôle. Du point de vue du *blossoming*, l'évolution du polygone de contrôle n'est qu'une conséquence de l'évolution du vecteur inter-nodal.

Les exemples quadratique et cubique sont d'abord présentés puis la méthode générique pour générer l'automate correspondant à une NUBS de degré quelconque est expliquée.

##### Automate CIFS de NUBS quadratiques

Le vecteur inter-nodal d'une NUBS quadratique d'un seul morceau est composé de 3 inter-noeuds :  $[v, w, x]$ . A partir de ce vecteur de départ, l'automate est construit par la méthode du dédoublement des inter-noeuds (voir Figure 4.12).

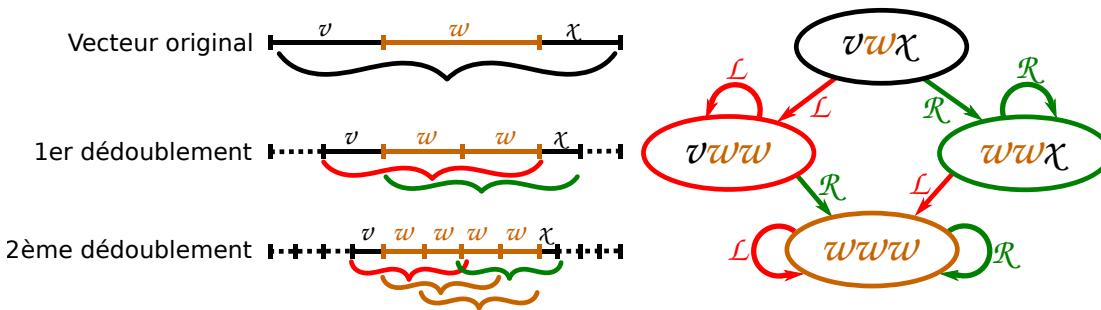


FIGURE 4.12 – Construction de l'automate CIFS de NUBS quadratiques. A gauche, l'évolution du vecteur inter-nodal. Tous les états possibles pour le vecteur inter-nodal sont atteints en deux dédoublements. A droite, l'automate correspondant est construit au fur et à mesure : chaque nouvel état du vecteur inter-nodal est ajouté à l'automate.

Il apparaît que 2 dédoublements sont nécessaires pour atteindre la totalité des séquences de 3 inter-noeuds consécutifs possibles. Chacune de ces possibilités correspond à un état de l'automate. Ces états sont au nombre de 4 et peuvent être classés comme ceci :

- **L'état de départ**, seul état avec 3 valeurs distinctes
- **L'état stationnaire à gauche** qui n'est donc pas modifié par l'application de  $L$
- **L'état stationnaire à droite** qui n'est donc pas modifié par l'application de  $R$
- **L'état uniforme** qui est stationnaire à gauche et à droite et par conséquent final

##### Automate CIFS de NUBS cubiques

Le vecteur inter-nodal d'une NUBS cubique d'un seul morceau est composé de 5 inter-noeuds :  $[x, v, w, x, y]$ . L'automate CIFS correspondant est donné en Figure 4.13.

Les 8 états de l'automate sont atteints en 3 dédoublements et peuvent être classés comme ceci :

- **L'état de départ**, seul état avec 5 valeurs distinctes
- **L'état stationnaire à gauche** qui n'est donc pas modifié par l'application de  $L$
- **L'état stationnaire à droite** qui n'est donc pas modifié par l'application de  $R$
- **L'état uniforme** qui est stationnaire à gauche et à droite et par conséquent final
- 4 autres états dits "de transition"

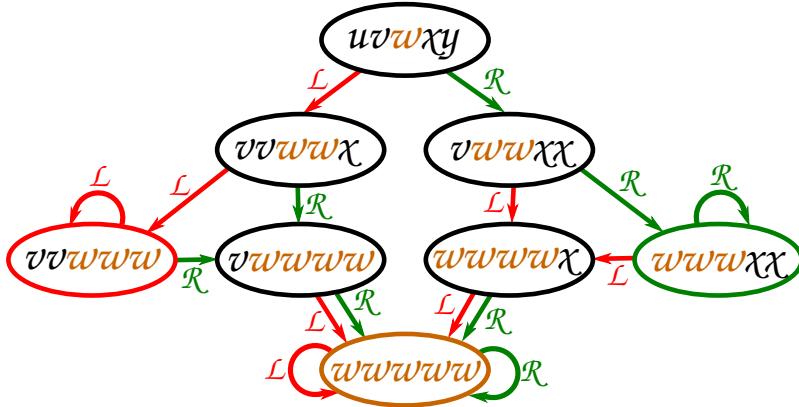


FIGURE 4.13 – Automate CIFS de NURBS cubique.

### Automate CIFS de NUBS de degré quelconque

En étudiant les automates CIFS des NUBS de bas degrés (inférieurs à 5), des généralisations apparaissent :

- Le nombre d'états de l'automate est fini et égal à  $4(d - 1)$
- Tous les états sont générés en  $\lceil \log_2(2d - 1) \rceil$  applications de  $\mathcal{L}$  ou  $\mathcal{R}$
- La valeur centrale de chaque vecteur est toujours conservée lors de l'application de  $\mathcal{L}$  et  $\mathcal{R}$
- Seul le vecteur inter-nodal uniforme  $W = [w \dots w]$  satisfait l'équation  $W = \mathcal{L}(W) = \mathcal{R}(W)$
- Ormis  $W$ , seul  $W_{\mathcal{L}} = [v^{(d-1)} w^d]$  est stationnaire à gauche ( $W_{\mathcal{L}} = \mathcal{L}(W_{\mathcal{L}})$ )
- Ormis  $W$ , seul  $W_{\mathcal{R}} = [w^d x^{(d-1)}]$  est stationnaire à droite ( $W_{\mathcal{R}} = \mathcal{R}(W_{\mathcal{R}})$ )
- Toute séquence de transformations finissant par une infinité de répétitions de  $\mathcal{L}$  (respectivement  $\mathcal{R}$ ) finira par boucler sur l'état  $W$  ou  $W_{\mathcal{L}}$  (respectivement  $W_{\mathcal{R}}$ )

Une démonstration proposée par Jean-Luc Baril est présente en [Annexe A](#).

Ces résultats assurent que quel que soit le degré de la NUBS, l'automate CIFS est constructible (nombre fini d'états) et exécutable en temps fini. En effet, toute adresse terminant par une infinité de répétition de la même transformation conduit à un des états stationnaires de cette transformation. Une fois cet état stationnaire atteint, il ne reste plus qu'à remplacer l'infinité de transformations identiques qui reste à appliquer par le point-fixe de cette transformation.

### 4.2.5 Automate CIFS de surfaces NUBS tensorielles de degré quelconque

L'automate CIFS correspondant au produit tensoriel de deux automates CIFS de NUBS peut être construit en utilisant la méthode décrite dans la [Sous-section 2.4.3 page 38](#). Pour cela, deux automates de courbe sont utilisés : un automate "horizontal" dont les transformations sont  $\mathcal{L}$  et  $\mathcal{R}$  et un automate "vertical" où les transformations sont renommées  $\mathcal{D}$  et  $\mathcal{U}$ . Les transformations de l'automate de surface sont donc :

$$\mathcal{T}_3 = \mathcal{U} \otimes \mathcal{L} \quad \mathcal{T}_2 = \mathcal{U} \otimes \mathcal{R}$$

$$\mathcal{T}_0 = \mathcal{D} \otimes \mathcal{L} \quad \mathcal{T}_1 = \mathcal{D} \otimes \mathcal{R}$$

Les [Figures 4.14 et 4.15](#) présentent respectivement les automates CIFS de NUBS tensorielles bi-quadratiques et bi-cubiques.

Les automates de surfaces ont les états notables suivants :

- 4 états stationnaires pour une unique transformation
- 4 états stationnaires pour deux transformations (ayant une sous-transformation en commun)
- 1 état final uniforme unique

Ceci est dû aux propriétés du produit tensoriel d'automate : la composition de deux états stationnaires crée un état stationnaire et celle de deux états finaux crée un état final.

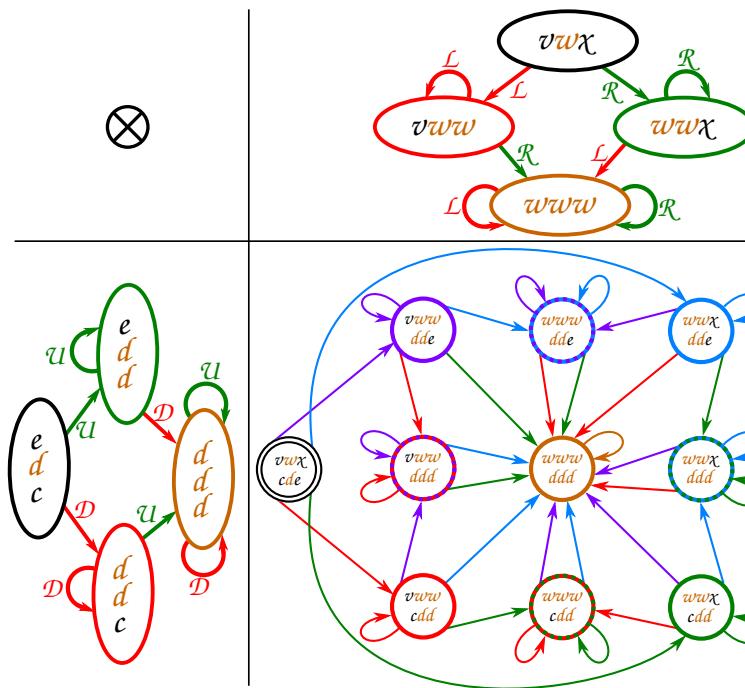


FIGURE 4.14 – Construction de l’automate CIFS de surfaces NUBS tensorielles bi-quadratiques. A gauche le vecteur inter-nodal "vertical", en haut le vecteur inter-nodal "horizontal", et au centre le résultat du produit tensoriel.

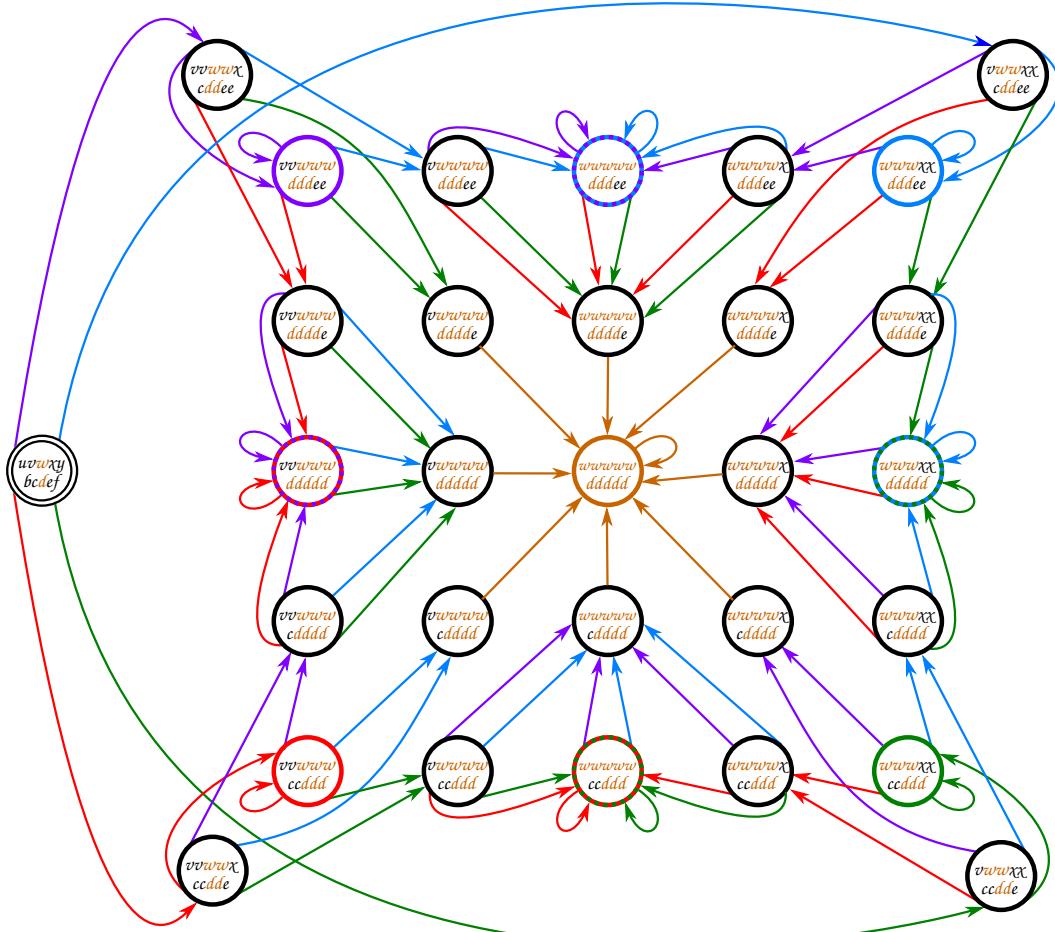


FIGURE 4.15 – Automate CIFS de surface NUBS tensorielles bi-cubiques.

## 4.3 Stratégie numéro 2 : uniformisation du vecteur nodal

Cette stratégie est inédite à ce manuscrit mais un article est en cours de rédaction [MGLN19]. L'idée derrière cette stratégie est d'exploiter au maximum le *blossoming*. Le *blossoming* est basé sur une notion d'étiquettes qui peuvent être exprimées en fonction d'autres étiquettes. Les points de contrôle ne sont qu'un cas particulier de ces étiquettes. La question est la suivante : est-il possible d'exprimer n'importe quelle étiquette en fonction du polygone de contrôle et par conséquent de trouver un polygone de contrôle qui définit une B-Spline uniforme correspondant à une NUBS quelconque ? Si c'est le cas, il suffit de transformer la NUBS en une B-Spline uniforme de même degré, puis d'utiliser lIFS des B-Splines uniformes [ZT96, SLG05] pour la calculer.

### 4.3.1 Expression d'une étiquette uniforme en fonction du polygone de contrôle

Jusqu'à présent, les étiquettes du nouveau polygone étaient choisies de manière à être incluses dans les anciennes. Par conséquent le nouveau vecteur nodal était lui aussi inclus dans l'ancien. Ceci assurait de retrouver les étiquettes correspondant aux points de contrôle en appliquant le *blossoming*. En commençant par les cas quadratique et cubique puis en généralisant au degré quelconque, une étiquette d'arguments quelconques est exprimée en fonction du polygone de contrôle. Si une étiquette quelconque peut être calculée, les étiquettes uniformes peuvent l'être aussi et par conséquent un polygone de contrôle uniforme décrivant la même courbe est constructible.

#### Etiquettes quadratiques

Soit une étiquette quadratique quelconque  $\{a, b\}$  et les étiquettes des points de contrôle  $p_0 : \{t_0, t_1\}$ ,  $p_1 : \{t_1, t_2\}$ , et  $p_2 : \{t_2, t_3\}$ . Le *blossoming* permettant l'expression de  $\{a, b\}$  en fonction des points de contrôle est donné en Figure 4.16.

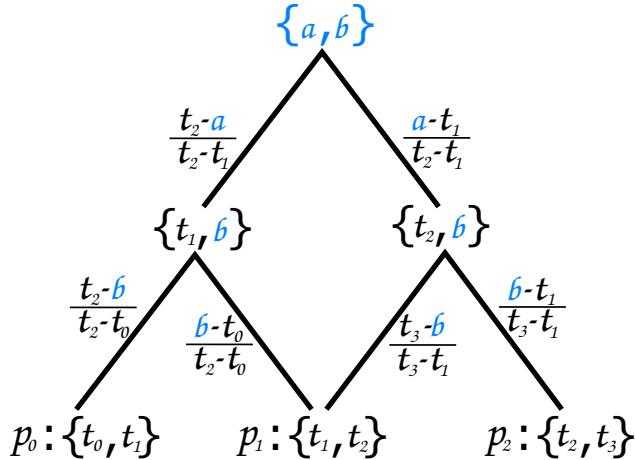
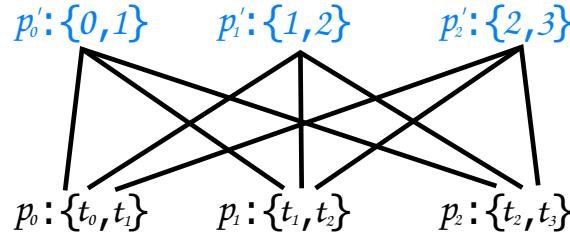


FIGURE 4.16 – *Blossoming* de l'uniformisation d'une NUBS quadratique

Le résultat obtenu est le suivant :

$$\{\a, b\} = \frac{t_2 - a}{t_2 - t_1} \frac{t_2 - b}{t_2 - t_0} p_0 + \left( \frac{t_2 - a}{t_2 - t_1} \frac{b - t_0}{t_2 - t_0} + \frac{a - t_1}{t_2 - t_1} \frac{t_3 - b}{t_3 - t_1} \right) p_1 + \frac{a - t_1}{t_2 - t_1} \frac{b - t_1}{t_3 - t_1} p_2$$

Il est donc possible d'exprimer les étiquettes uniformes  $p'_0 = \{0, 1\}$ ,  $p'_1 = \{1, 2\}$ , et  $p'_2 = \{2, 3\}$  correspondant au vecteur uniforme  $\mathbf{T} = [0, 1, 2, 3]$  en fonction du polygone de contrôle original (voir Figure 4.17). *N.B.* : les nouveaux points ne sont dans l'enveloppe convexe des anciens que dans le cas où  $t_1 \leq a \leq b \leq t_2$ , sinon il y a apparition de coefficients négatifs dans le calcul. La présence de coefficients négatifs n'a pas de conséquence sur la construction du *blossoming* hormis la sortie de cette enveloppe convexe (au même titre qu'une combinaison barycentrique à coefficient négatif).

FIGURE 4.17 – Uniformisation d'une NUBS quadratique par *blossoming*

Tout point de la nouvelle B-Spline  $\mathcal{C}'(t)$  est calculé grâce aux coefficients uniformes :

$$\mathcal{C}'(t) = \frac{(2-t)^2}{2} p'_0 + \frac{-2t^2 + 6t - 3}{2} p'_1 + \frac{(t-1)^2}{2} p'_2$$

Il faut maintenant vérifier que les deux polygones définissent bien la même courbe. Autrement dit, il faut prouver que le *blossoming* en Figure 4.18 ne modifie pas la courbe.

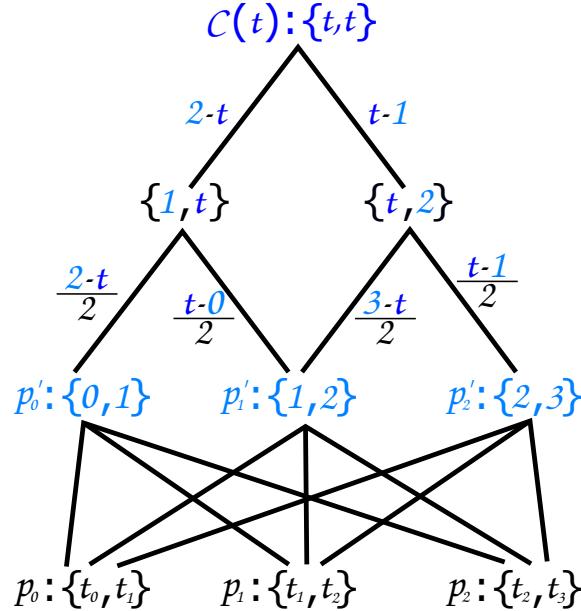


FIGURE 4.18 – *Blossoming* exprimant un point de la NUBS en fonction des points de contrôle en passant par un polygone de contrôle intermédiaire uniforme. La ligne du bas est le polygone de contrôle de la NUBS et la ligne cyan est le nouveau polygone de contrôle uniforme. La NUBS  $\mathcal{C}(t)$  repose à la fois sur le polygone original avec des coefficients non-uniformes et sur le nouveau polygone avec des coefficients uniformes.

Pour cela, il faut prouver que les influences  $\mathcal{I}(p_i)$  des points du polygone de contrôle de départ sur la B-Spline uniforme  $\mathcal{C}'(t)$  sont bien les mêmes que celles sur la NUBS  $\mathcal{C}(t)$ .

Pour rappel ces influences sont :

$$N_j^d(t) = \frac{t - t_j}{t_{(j+d)} - t_j} N_j^{(d-1)}(t) + \frac{t_{(j+d+1)} - t}{t_{(j+d+1)} - t_{(j+1)}} N_{(j+1)}^{(d-1)}(t)$$

$$N_j^0(t) = 1 \text{ si } t \in [t_j; t_{j+1}[ \text{ et } 0 \text{ sinon}$$

$$\begin{aligned}
\mathcal{I}(p_0) &= \frac{(2-\textcolor{blue}{t})^2}{2} \frac{t_2-0}{t_2-t_1} \frac{t_2-1}{t_2-t_0} + \frac{-2\textcolor{blue}{t}^2+6\textcolor{blue}{t}-3}{2} \frac{t_2-1}{t_2-t_1} \frac{t_2-2}{t_2-t_0} + \frac{(\textcolor{blue}{t}-1)^2}{2} \frac{t_2-2}{t_2-t_1} \frac{t_2-3}{t_2-t_0} \\
2(t_2-t_1)(t_2-t_0)\mathcal{I}(p_0) &= [t_2^2\textcolor{blue}{t}^2 - 4t_2^2\textcolor{blue}{t}^2 + 4t_2^2 - t_2\textcolor{blue}{t}^2 + 4t_2\textcolor{blue}{t} - 4t_2] \\
&\quad + [-2t_2^2\textcolor{blue}{t}^2 + 6t_2^2\textcolor{blue}{t} - 3t_2^2 + 6t_2\textcolor{blue}{t}^2 - 18t_2\textcolor{blue}{t} + 9t_2 - 4\textcolor{blue}{t}^2 + 12\textcolor{blue}{t} - 6] \\
&\quad + [t_2^2\textcolor{blue}{t}^2 - 2t_2^2\textcolor{blue}{t} + t_2^2 - 5t_2\textcolor{blue}{t}^2 + 10t_2\textcolor{blue}{t} - 5t_2 + 6\textcolor{blue}{t}^2 - 12\textcolor{blue}{t} + 6] \\
&= t_2^2\textcolor{blue}{t}^2(1-2+1) + t_2^2\textcolor{blue}{t}(-4+6-2) + t_2^2(4-3+1) + t_2\textcolor{blue}{t}^2(-1+6-5) \\
&\quad + t_2\textcolor{blue}{t}(4-18+10) + t_2(-4+9-5) + \textcolor{blue}{t}^2(-4+6) + \textcolor{blue}{t}(12-12) + (-6+6) \\
&= 2t_2^2 - 4t_2\textcolor{blue}{t} + 2\textcolor{blue}{t}^2 \\
&= 2(t_2-\textcolor{blue}{t})^2 \\
\mathcal{I}(p_0) &= \frac{(t_2-\textcolor{blue}{t})^2}{(t_2-t_1)(t_2-t_0)} \\
&= N_0^2(t)
\end{aligned}$$

$$\begin{aligned}
\mathcal{I}(p_2) &= \frac{(2-\textcolor{blue}{t})^2}{2} \frac{0-t_1}{t_2-t_1} \frac{1-t_1}{t_3-t_1} + \frac{-2\textcolor{blue}{t}^2+6\textcolor{blue}{t}-3}{2} \frac{1-t_1}{t_2-t_1} \frac{2-t_1}{t_3-t_1} + \frac{(\textcolor{blue}{t}-1)^2}{2} \frac{2-t_1}{t_2-t_1} \frac{3-t_1}{t_3-t_1} \\
2(t_2-t_1)(t_3-t_1)\mathcal{I}(p_2) &= [t_1^2\textcolor{blue}{t}^2 - 4t_1^2\textcolor{blue}{t} + 4t_1^2 - t_1\textcolor{blue}{t}^2 + 4t_1\textcolor{blue}{t} - 4t_1] \\
&\quad + [-2t_1^2\textcolor{blue}{t}^2 + 6t_1^2\textcolor{blue}{t} - 3t_1^2 + 6t_1\textcolor{blue}{t}^2 - 18t_1\textcolor{blue}{t} + 9t_1 - 4\textcolor{blue}{t}^2 + 12\textcolor{blue}{t} - 6] \\
&\quad + [t_1^2\textcolor{blue}{t}^2 - 2t_1^2\textcolor{blue}{t} + t_1^2 - 5t_1\textcolor{blue}{t}^2 + 10t_1\textcolor{blue}{t} - 5t_1 + 6\textcolor{blue}{t}^2 - 12\textcolor{blue}{t} + 6] \\
&= t_1^2\textcolor{blue}{t}^2(1-2+1) + t_1^2\textcolor{blue}{t}(-4+6-2) + t_1^2(4-3+1) + t_1\textcolor{blue}{t}^2(-1+6-5) \\
&\quad + t_1\textcolor{blue}{t}(4-18+10) + t_1(-4+9-5) + \textcolor{blue}{t}^2(-4+6) + \textcolor{blue}{t}(12-12) + (-6+6) \\
&= 2t_1^2 - 4t_1\textcolor{blue}{t} + 2\textcolor{blue}{t}^2 \\
&= 2(\textcolor{blue}{t}-t_1)^2 \\
\mathcal{I}(p_2) &= \frac{(\textcolor{blue}{t}-t_1)^2}{(t_2-t_1)(t_3-t_1)} \\
&= N_2^2(t)
\end{aligned}$$

Le calcul de  $\mathcal{I}(p_1)$  est conséquent mais peut être découpé en deux parties indépendantes  $\mathcal{I}'(p_1)$  et  $\mathcal{I}''(p_1)$

$$\begin{aligned}
\mathcal{I}'(p_1) &= \frac{(2-\textcolor{blue}{t})^2}{2} \frac{t_2-0}{t_2-t_1} \frac{1-t_0}{t_2-t_0} + \frac{-2\textcolor{blue}{t}^2+6\textcolor{blue}{t}-3}{2} \frac{t_2-1}{t_2-t_1} \frac{2-t_0}{t_2-t_0} + \frac{(\textcolor{blue}{t}-1)^2}{2} \frac{t_2-2}{t_2-t_1} \frac{3-t_0}{t_2-t_0} \\
2(t_2-t_1)(t_2-t_0)\mathcal{I}'(p_1) &= [-t_0t_2\textcolor{blue}{t}^2 + 4t_0t_2\textcolor{blue}{t} - 4t_0t_2 + t_2\textcolor{blue}{t}^2 - 4t_2\textcolor{blue}{t} + 4t_2] \\
&\quad + [2t_0t_2\textcolor{blue}{t}^2 - 6t_0t_2\textcolor{blue}{t} + 3t_0t_2 - 2t_0\textcolor{blue}{t}^2 + 6t_0\textcolor{blue}{t} - 3t_0 - 4t_2\textcolor{blue}{t}^2 + 12t_2\textcolor{blue}{t} - 6t_2 + 4\textcolor{blue}{t}^2 - 12\textcolor{blue}{t} + 6] \\
&\quad + [-t_0t_2\textcolor{blue}{t}^2 + 2t_0t_2\textcolor{blue}{t} - t_0t_2 + 2t_0\textcolor{blue}{t}^2 - 4t_0\textcolor{blue}{t} + 2t_0 + 3t_2\textcolor{blue}{t}^2 - 6t_2\textcolor{blue}{t} + 3t_2 - 6\textcolor{blue}{t}^2 + 12\textcolor{blue}{t} - 6] \\
&= t_0t_2\textcolor{blue}{t}^2(-1+2-1) + t_0t_2\textcolor{blue}{t}(4-6+2) + t_0t_2(-4+3-1) + t_0\textcolor{blue}{t}^2(-2+2) + t_0\textcolor{blue}{t}(6-4) + t_0(-3+2) \\
&\quad + t_2\textcolor{blue}{t}^2(1-4+3) + t_2\textcolor{blue}{t}(-4+12-6) + t_2(4-6+3) + \textcolor{blue}{t}^2(-4+6) + \textcolor{blue}{t}(-12+12) + (6-6) \\
&= -2t_0t_2 + 2t_0\textcolor{blue}{t} - t_0 + 2t_2\textcolor{blue}{t} + t_2 - 2\textcolor{blue}{t}^2 \\
&= 2(t_2-\textcolor{blue}{t})(\textcolor{blue}{t}-t_0) + t_2 - t_0 \\
\mathcal{I}'(p_1) &= \frac{(t_2-\textcolor{blue}{t})(\textcolor{blue}{t}-t_0)}{(t_2-t_1)(t_2-t_0)} + \frac{t_2-t_0}{2(t_2-t_1)(t_2-t_0)} \\
&= \frac{(t_2-\textcolor{blue}{t})(\textcolor{blue}{t}-t_0)}{(t_2-t_1)(t_2-t_0)} + \frac{1}{2(t_2-t_1)}
\end{aligned}$$

$$\begin{aligned}
\mathcal{I}''(p_1) &= \frac{(2 - \textcolor{blue}{t})^2}{2} \frac{0 - t_1}{t_2 - t_1} \frac{t_3 - 1}{t_3 - t_1} + \frac{-2\textcolor{blue}{t}^2 + 6\textcolor{blue}{t} - 3}{2} \frac{1 - t_1}{t_2 - t_1} \frac{t_3 - 2}{t_3 - t_1} + \frac{(\textcolor{blue}{t} - 1)^2}{2} \frac{2 - t_1}{t_2 - t_1} \frac{t_3 - 3}{t_3 - t_1} \\
2(t_2 - t_1)(t_3 - t_1)\mathcal{I}''(p_1) &= [-t_1 t_3 \textcolor{blue}{t}^2 + 4t_1 t_3 \textcolor{blue}{t} - 4t_1 t_3 + t_1 \textcolor{blue}{t}^2 - 4t_1 \textcolor{blue}{t} + 4t_1] \\
&\quad + [2t_1 t_3 \textcolor{blue}{t}^2 - 6t_1 t_3 \textcolor{blue}{t} + 3t_1 t_3 - 4t_1 \textcolor{blue}{t}^2 + 12t_1 \textcolor{blue}{t} - 6t_1 - 2t_3 \textcolor{blue}{t}^2 + 6t_3 \textcolor{blue}{t} - 3t_3 + 4\textcolor{blue}{t}^2 - 12\textcolor{blue}{t} + 6] \\
&\quad + [-t_1 t_3 \textcolor{blue}{t}^2 + 2t_1 t_3 \textcolor{blue}{t} - t_1 t_3 + 3t_1 \textcolor{blue}{t}^2 - 6t_1 \textcolor{blue}{t} + 3t_1 + 2t_3 \textcolor{blue}{t}^2 - 4t_3 \textcolor{blue}{t} + 2t_3 - 6\textcolor{blue}{t}^2 + 12\textcolor{blue}{t} - 6] \\
&= t_1 t_3 \textcolor{blue}{t}^2 (-1 + 2 - 1) + t_1 t_3 \textcolor{blue}{t} (4 - 6 + 2) + t_1 t_3 (-4 + 3 - 1) + t_1 \textcolor{blue}{t}^2 (1 - 4 + 3) + t_1 \textcolor{blue}{t} (-4 + 12 - 6) \\
&\quad + t_1 (4 - 6 + 3) + t_3 \textcolor{blue}{t}^2 (-2 + 2) + t_3 \textcolor{blue}{t} (6 - 4) + t_3 (-3 + 2) + \textcolor{blue}{t}^2 (4 - 6) + \textcolor{blue}{t} (-12 + 12) + (6 - 6) \\
&= -2t_1 t_3 + 2t_1 \textcolor{blue}{t} + t_1 + 2t_3 \textcolor{blue}{t} - t_3 - 2\textcolor{blue}{t}^2 \\
&= 2(t_3 - \textcolor{blue}{t})(\textcolor{blue}{t} - t_1) + t_1 - t_3 \\
\mathcal{I}''(p_1) &= \frac{(t_3 - \textcolor{blue}{t})(\textcolor{blue}{t} - t_1)}{(t_2 - t_1)(t_3 - t_1)} + \frac{t_1 - t_3}{2(t_2 - t_1)(t_3 - t_1)} \\
&= \frac{(t_3 - \textcolor{blue}{t})(\textcolor{blue}{t} - t_1)}{(t_2 - t_1)(t_3 - t_1)} + \frac{-1}{2(t_2 - t_1)}
\end{aligned}$$

$$\begin{aligned}
\mathcal{I}(p_1) &= \mathcal{I}'(p_1) + \mathcal{I}''(p_1) \\
&= \frac{(t_2 - \textcolor{blue}{t})(\textcolor{blue}{t} - t_0)}{(t_2 - t_1)(t_2 - t_0)} + \frac{(t_3 - \textcolor{blue}{t})(\textcolor{blue}{t} - t_1)}{(t_2 - t_1)(t_3 - t_1)} + \frac{1}{2(t_2 - t_1)} + \frac{-1}{2(t_2 - t_1)} \\
&= \frac{(t_2 - \textcolor{blue}{t})(\textcolor{blue}{t} - t_0)}{(t_2 - t_1)(t_2 - t_0)} + \frac{(t_3 - \textcolor{blue}{t})(\textcolor{blue}{t} - t_1)}{(t_2 - t_1)(t_3 - t_1)} \\
&= N_1^2(t)
\end{aligned}$$

Les influences obtenues par le calcul sont bien celles des NUBS quadratiques classiques.

### Etiquettes cubiques

De la même manière que les NUBS quadratiques, les étiquettes cubiques quelconques peuvent être exprimées en fonction du polygone de contrôle (voir Figure 4.19). Il est donc possible d'uniformiser n'importe quelle NUBS cubique à un seul morceau.

### Etiquettes de degré quelconque

A partir des degrés 2 et 3, il est facile d'imaginer la généralisation au degré quelconque. Comme l'étiquette composée de  $d$  arguments identiques  $\{\textcolor{blue}{t} \dots \textcolor{blue}{t}\}$  est calculable par un *blossoming* de  $(d + 1)$  lignes, les étiquettes quelconques  $\{a \dots z\}$  le sont également.

*N.B.* : les bornes sont conservées *i.e.*  $\textcolor{blue}{t}$  varie de  $t_{(d-1)}$  à  $t_d$  à la fois pour la NUBS  $\mathcal{C}$  et pour la B-Spline uniforme correspondante  $\mathcal{C}'$ . Du point de vue du *blossoming*, cela se traduit par la réflexion suivante : les branches peuvent être réorganisées de manière quelconque sans modifier la courbe obtenue tant que la racine et les feuilles sont inchangées.

Dans le calcul d'une B-Spline uniforme de degré  $d$ , il faut se donner un standard pour pouvoir pré-calculer les influences des points (par exemple de toujours faire varier  $t$  entre  $(d - 1)$  et  $d$ ). Pour cela, le vecteur nodal de  $\mathcal{C}$  subit un changement de bornes préalable pour que la courbe soit bien définie entre  $(d - 1)$  et  $d$  :

$$t_i \leftarrow \frac{t_i - t_{(d-1)}}{t_d - t_{(d-1)}} + (d - 1)$$

Le changement de bornes ne modifie pas la courbe car les mises à l'échelle et les translations n'ont pas d'influence sur le vecteur nodal. Il ne reste plus qu'à appliquer la méthode de calcul des étiquettes uniformes par *blossoming* et de tracer la courbe en faisant varier  $t \in [(d - 1); d]$

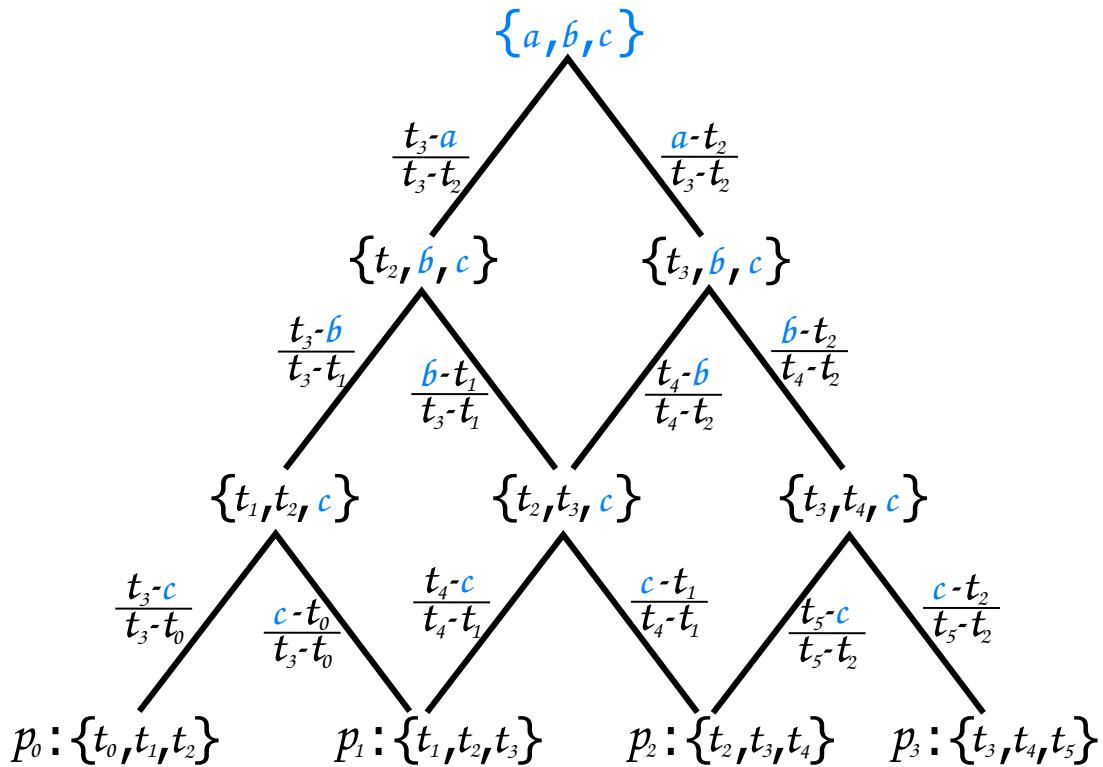


FIGURE 4.19 – Expression d'une étiquette cubique quelconque en fonction du polygone de contrôle par *blossoming*.

Ceci implique que dans l'uniformisation d'une NUBS composée de plusieurs morceaux, chaque morceau doit être uniformisé indépendamment des autres, ce qui mène à la construction d'autant de polygones uniformes que de morceaux (voir Figure 4.20).

### Résumé de la méthode

Premièrement, les différents morceaux de la NUBS sont séparés en autant de sous-polygones de contrôle et de sous-vecteurs nodaux. Chaque sous-vecteur nodal est ensuite normalisé pour que tous les morceaux de courbe soient définis entre les bornes ( $d - 1$ ) et  $d$ .

Ensuite les  $(d+1)$  étiquettes uniformes  $p'_0 : \{0 \dots (d-1)\} \dots p'_d : \{d \dots (2d-1)\}$  sont calculées par *blossoming* pour trouver l'influence des points de contrôle  $p_0 \dots p_d$  sur chaque étiquette en fonction du vecteur nodal  $[t_0 \dots t_{2d-1}]$ . Ces influences sont appliquées sur chaque morceau de courbe pour construire le polygone de contrôle uniforme correspondant.

Enfin, chaque polygone de contrôle permet la construction d'un IFS de B-Spline uniforme [SLG05] (cf. Sous-section 2.3.6) dont la courbe limite est directement calculable et correspond à un morceau de la NUBS.

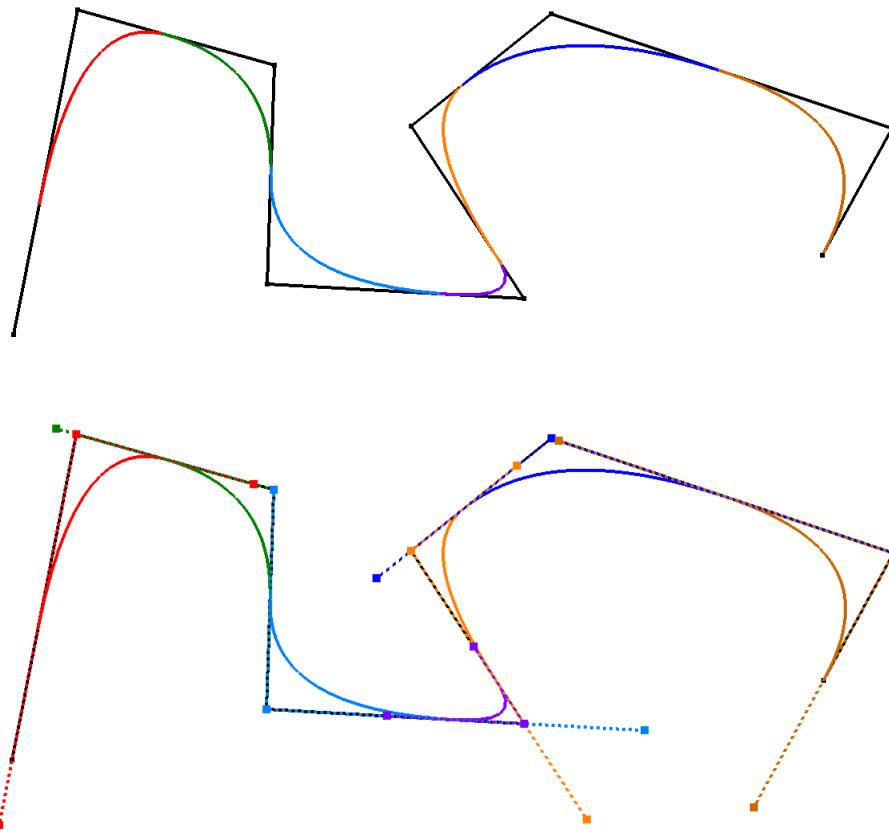


FIGURE 4.20 – Uniformisation d'une NUBS quadratique à sept morceaux. En haut, une NUBS quadratique dont le vecteur nodal est  $\mathbf{T} = [0, 0.6, 1.5, 2.6, 3.7, 4.1, 5.5, 7.8, 10]$ . Le polygone de contrôle est en noir et chaque morceau de la courbe limite est affiché avec une couleur différente. En bas, la même courbe mais où chaque morceau est uniforme. Le polygone de contrôle de chaque morceau est de la même couleur que le morceau qu'il décrit. Le fait que le point de contrôle central de chaque polygone uniforme est exactement celui du polygone original n'est qu'un cas particulier dû au degré 2, il n'est pas valable pour les degrés plus élevés.

### 4.3.2 Uniformisation de surfaces NUBS tensorielles

Une surface NUBS tensorielle possède deux vecteurs nodaux (un pour chaque courbe) qui peuvent correspondre à des NUBS de degré différents :  $d$  et  $d'$ . Les points de contrôle de la surface sont représentés en *blossoming* par une **étiquette-double** dont chaque partie est indépendante de l'autre et n'agit que sur un seul des vecteurs nodaux. Le *blossoming* peut être calculé de deux manières différentes :

- En deux étapes : la première partie de l'étiquette-double est découpée en  $(d + 1)$  étiquettes-doubles puis la deuxième partie de chacune de ces étiquettes est à nouveau découpée en  $(d' + 1)$  étiquettes-doubles correspondant aux points de contrôle
- Directement en découplant les deux parties d'une étiquette-double en même temps créant ainsi quatre étiquettes-doubles. Les coefficients obtenus sont les produit des coefficients du découpage de chaque partie.

Dans les deux cas, les points de contrôle sont organisés sur une grille de taille  $(d + 1) \times (d' + 1)$  comme dans la construction classique d'une surface tensorielle à partir de deux NUBS. La seconde sera privilégiée car plus compacte (moins d'opérations même si celles-ci sont plus "difficiles"). Un exemple est donné pour une surface NUBS tensorielle bi-quadratique en Figure 4.21.

Comme visible dans la Figure 4.21, n'importe quelle étiquette-double peut être exprimée en fonction des points de contrôle. Il est donc possible de trouver une grille de contrôle dont la surface B-Spline tensorielle bi-quadratique correspond exactement à un morceau de surface NUBS tensorielle bi-quadratique. D'une manière plus générale, toute surface NUBS tensorielle peut être uniformisée avec une grille de contrôle pour chaque morceau de surface.

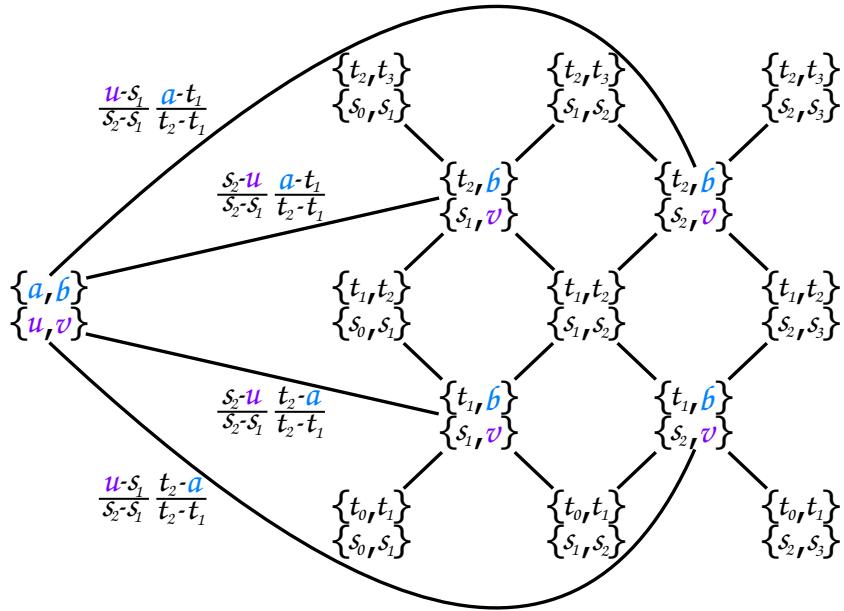


FIGURE 4.21 – Expression d'un étiquette-double biquadratique quelconque par *blossoming*. Seuls les coefficients du premier découpage sont indiqués pour des raisons de lisibilité.

### En résumé :

- Les NURBS sont la représentation de courbes par morceaux la plus utilisée en CGAO. Elles sont définies par un polygone de contrôle et un vecteur nodal.
- En représentant les différents états du vecteur nodal par des états de l'automate, un CIFS peut être construit quelque soit le degré de la NURBS.
- Les NURBS peuvent également être uniformisées pour devenir des B-Splines uniformes et être alors construites par un IFS barycentrique.
- Ces deux méthodes de représentation des NURBS sont extensibles aux surfaces NURBS tensorielles par produit-tensoriel.

# Chapitre 5

# Applications

Jusqu'à présent, le manuscrit ne contient que des contributions théoriques. Dans ce chapitre, sont présentées deux applications qui découlent de ces contributions.

La première application est un moteur de rendu temps réel de CIFS barycentriques, appelé *shaders* de combinaisons barycentriques. Ce moteur de rendu exploite la puissance des cartes graphiques récentes et le *pipeline* programmable de l'OpenGL moderne (version minimum requise : 4.3 de 2012 pour OpenGL et GLSL) et en particulier les *Tesselation Shaders*. Grâce à cette nouvelle architecture, les surfaces de subdivision à support compact, les surfaces tensorielles B-Splines uniformes ou NURBS, ainsi que les surfaces fractales non-lacunaires peuvent être calculées et affichées à la volée à partir de patches du maillage de contrôle envoyés sur la carte graphique et de combinaisons barycentriques pré-calculées. Cette architecture a été implémentée dans le logiciel Subdiv, entièrement développé pour les besoins de la thèse.

La seconde application est l'ajout des automates CIFS de surfaces de subdivision et de NURBS au modeleur géométrique de CGAO : MODITERE (© Laboratoire d'Informatique de Bourgogne). Ce modeleur, maintenu par Gilles Gouaty, utilise la représentation sous forme d'automates CIFS des surfaces et des volumes pour l'optimisation topologique.

## 5.1 *Shaders* de combinaisons barycentriques pour le rendu temps réel

Dans cette section est décrite l'architecture d'un moteur de rendu temps réel des automates CIFS barycentriques. Ce moteur de rendu permet un compromis entre deux méthodes plus traditionnelles de rendu de surfaces de subdivision : la **subdivision préalable** qui consomme beaucoup de mémoire car il faut envoyer le maillage le plus affiné (et donc le plus lourd) sur la carte graphique et la **subdivision par règles** sur la carte graphique qui nécessite plusieurs passes et donc est long en calcul. La méthode présentée ici permet de calculer les surfaces de subdivision à partir de patches du maillage de contrôle (donc moins lourd que la subdivision préalable) en une seule passe sur la carte graphique (donc plus rapide que la subdivision par règles). Même si cette architecture a principalement été utilisée pour le rendu de surfaces de subdivision, comme elle est basée sur la notion d'automates CIFS, d'autres types de surfaces peuvent également être implémentées pour le rendu temps réel.

Après avoir décrit le *pipeline* graphique de l'OpenGL moderne, il sera présenté comment celui-ci peut être exploité (et en particulier la phase de tessellation) pour permettre un rendu à la volée d'automate CIFS à partir de patch du maillage de contrôle. Ensuite, la méthode sera comparée aux deux méthodes classiquement utilisées dans le rendu de surfaces de subdivision. Pour finir, les limitations du moteur seront discutées et des pistes d'améliorations seront proposées. Le moteur présenté ici est une version plus détaillée que celle publiée à la conférence internationale *World Symposium of Computer Graphics* de 2018 [[MNLG18b](#)].

## Pipeline OpenGL du logiciel Subdiv

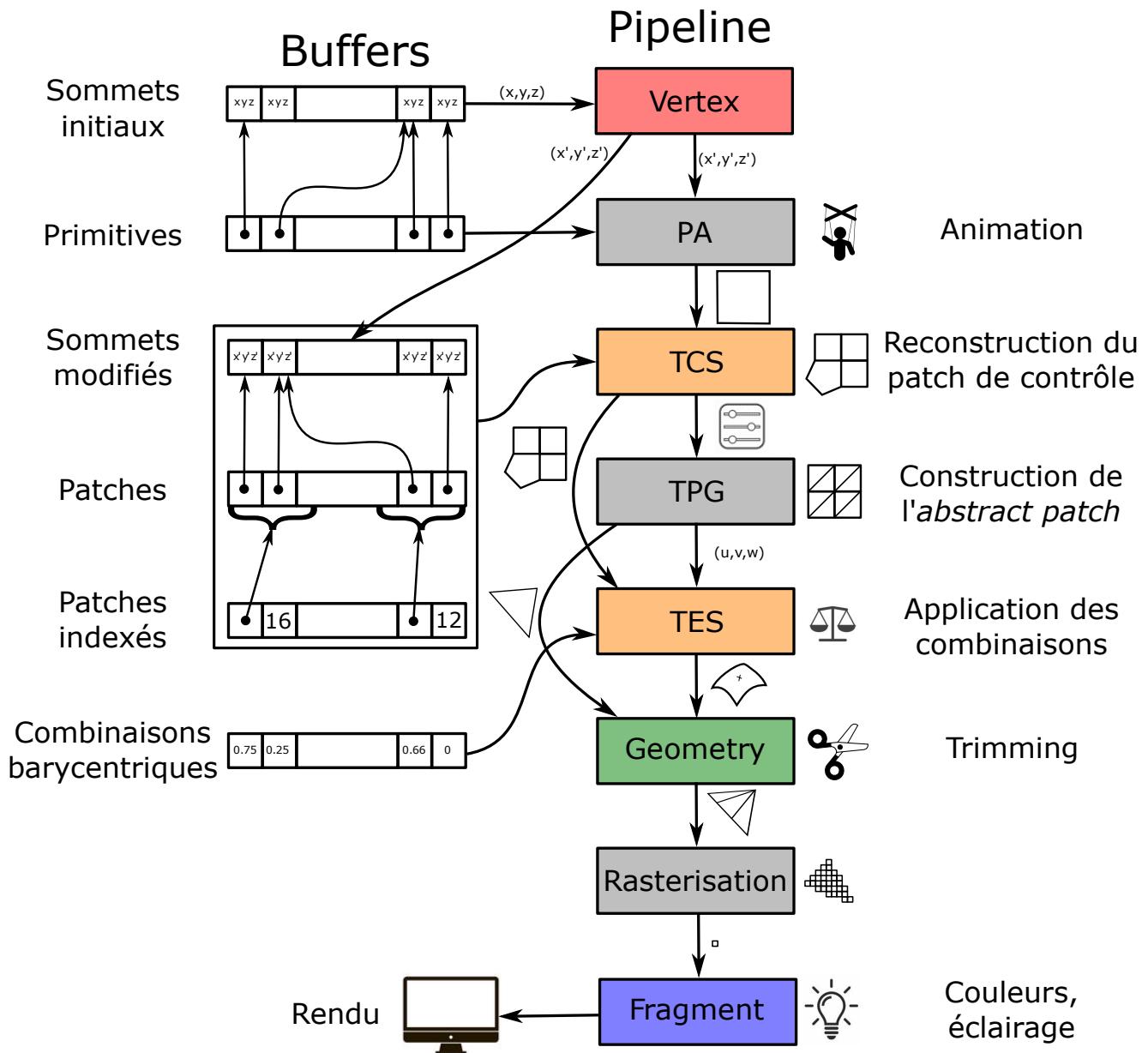


FIGURE 5.1 – Schéma du *pipeline OpenGL* dans le moteur de rendu par *shaders de combinaisons barycentriques*. Les étapes grises sont les étapes non-programmables du *pipeline*. Les deux premiers *buffers* sont ceux de base dans OpenGL et les quatre autres ont été ajoutés pour l'implémentation.

### 5.1.1 Le *pipeline* des *shaders* en OpenGL

En OpenGL moderne, le *pipeline* de rendu est composé de plusieurs étapes non-programmables et de 5 étapes programmables appelée *shaders*. Ces *shaders* sont exécutés dans l'ordre suivant :

- 1 - le *Vertex Shader* qui agit sur les sommets
- 2 - le *Tesselation Control Shader* qui contrôle le niveau de tessellation
- 3 - le *Tesselation Evaluation Shader* qui calcule la tessellation
- 4 - le *Geometry Shader* qui agit sur les faces
- 5 - le *Fragment Shader* qui s'occupe de la couleur et la lumière

Il y a également un autre shader : le *Compute Shader* qui ne fait pas partie du *pipeline* et sert à faire des calculs qui n'ont pas forcément de lien avec le rendu, mais il ne servira pas dans le moteur de rendu décrit ici. Les cinq autres font l'objet des sous-sections suivantes. Pour plus de clarté, il est conseillé de lire ces sous-sections tout en se reportant à la [Figure 5.1](#).

#### Le *Vertex Shader*

Comme son nom l'indique, c'est le *shader* qui s'occupe des sommets. Le *Vertex Shader* va se dupliquer en autant d'exemplaires qu'il y a de sommets en entrée et chaque exemplaire va traiter un sommet indépendamment des autres (dont il ignore même l'existence). Le *Vertex Shader* prend au minimum un sommet en entrée et renvoie un sommet. Un sommet est au minimum composé d'un vecteur de 4 coordonnées indiquant la position du sommet (`vec4 gl_Position`) auquel peuvent être ajoutées des informations de couleurs, de normales, de textures... La composition d'un sommet doit être la même pour tous les sommets en entrée et tous les sommets en sortie mais elle n'est pas forcément cohérente entre l'entrée et la sortie (*e.g.* un *Vertex Shader* qui attribue une couleur à un sommet en fonction de sa position aura une information de plus en sortie qu'en entrée). Il est également possible d'avoir des valeurs auxquelles chaque *Vertex Shader* a accès en lecture, mais pas en écriture (car il est impossible de savoir dans quel ordre ils sont exécutés). Ces valeurs servent par exemple à synchroniser sur une horloge commune les sommets pour une animation.

#### L'étape d'assemblage des primitives

Cette étape appelée *Primitive assembly* est toujours exécutée après le *Vertex Shader*. Selon l'étape programmable qui va la suivre, elle ne construit pas le même type de primitives :

- Si elle est suivie par le *Fragment Shader*, elle construit des triangles à rasteriser
- Si elle est suivie par le *Geometry Shader*, elle construit les primitives définies par l'utilisateur dans le *buffer* de primitives (`GL_ELEMENT_ARRAY_BUFFER`)
- Si elle est suivie par l'étape de tessellation, elle construit les patches qui seront utilisés dans cette étape.

Les sommets constituant cette primitive sont ceux en sortie du *Vertex Shader*.

#### L'étape de tessellation

Souvent appelé par abus de langage le *Tessellation Shader*, l'étape de tessellation du *pipeline* est en réalité composée de deux *shaders* et d'une étape non-programmable.

Pour commencer, un *Tessellation Control Shader* est exécuté pour chaque patch reçu du *Primitive Assembly* et autant de fois qu'il y a de sommets dans ce patch. Chaque exécution est chargée de renseigner son sommet courant et toutes les exécutions concernant le même patch peuvent écrire dans les variables communes de ce patch, en particulier les variables définissant le niveau de tessellation. Pour limiter les calculs, seule la première exécution de chaque patch (`gl_InvocationID = 0`) est chargée de renseigner les variables de patch et toutes les autres ne s'occupent que de leur sommet.

La tessellation peut être constante dans la totalité du rendu d'un maillage ou recalculée à chaque face et est définie sous la forme d'un *abstract patch*. Il existe deux types d'*abstract patches* : quadrangulaire et triangulaire. Chacun est défini comme un ensemble de niveau de subdivision des arêtes (`gl_TessLevelOuter[0...n]` avec  $n = 3$  pour les quadrangles et  $n = 2$  pour les triangles) et un ensemble de niveau de subdivision intérieur (`gl_TessLevelInner[i]` unique pour les triangles et double pour les quadrangles).

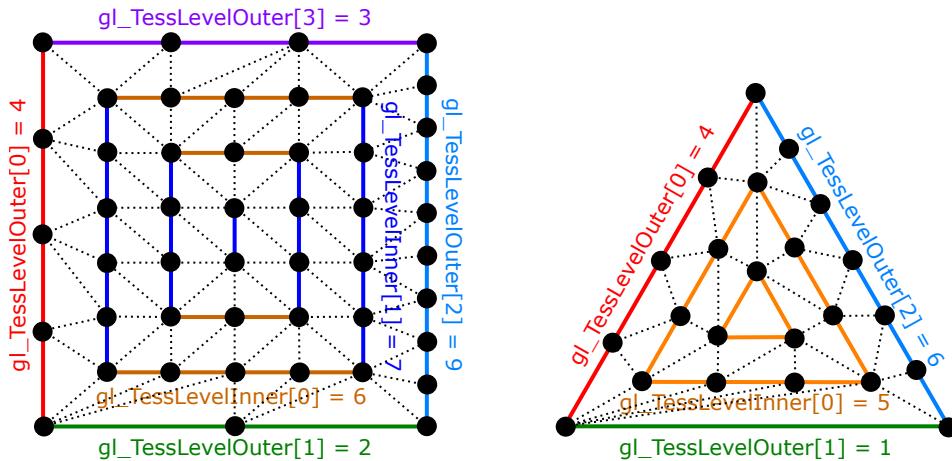


FIGURE 5.2 – Un exemple d'*abstract patch* quadrangulaire et un exemple triangulaire.

Ensuite, ces informations sont envoyées au *Tessellation Primitive Generator* qui va construire les primitives (des triangles) de cet *abstract patch*. Des exemples de construction sont donnés en Figure 5.2. Une fois les primitives construites, le *Tessellation Primitive Generator* envoie la liste des coordonnées de texture ( $u, v, w$ ) des sommets de l'*abstract patch* aux différents *Tessellation Evaluation Shader*.

Pour chaque face (ou patch) définie dans le maillage, et pour chaque sommet de l'*abstract patch* correspondant à cette face (ou ce patch), un *Tessellation Evaluation Shader* est exécuté. Celui-ci reçoit les coordonnées de textures du sommet courant de l'*abstract patch* ( $\text{vec3 } \text{gl\_TessCoord}$ ) et les sommets de la face (ou du patch) en cours de calcul ( $\text{gl\_PerVertex}[i]$ ). A partir de ces informations, il est chargé de renseigner le sommet correspondant obtenu par tesselation (au minimum la position du sommet,  $\text{vec4 } \text{gl\_Position}$ , doit être renseignée). Ce nouveau sommet fait alors partie de la primitive générée par le *Tessellation Primitive Generator*.

### Le *Geometry Shader*

Un *Geometry Shader* est exécuté pour chaque primitive (dont la topologie est reçue du *Tessellation Primitive Generator* et la position des sommets du *Tessellation Evaluation Shader*). Il y a plusieurs types de primitives définis en GLSL mais dans le cas où il y a eu une étape de tesselation, il s'agit forcément de triangles. Pour chaque primitive le *Geometry Shader* va générer une, plusieurs, ou même aucune primitive. Le nombre de primitives en sortie n'est pas limité mais le nombre de sommets l'est par la constante `GL_MAX_GEOMETRY_OUTPUT_VERTICES` (1024 sur les implémentations de GLSL les plus faibles).

Le *Geometry Shader* est principalement utilisé pour l'application de la **matrice MVP**. Cette matrice est le produit de trois matrices (**MVP** = **Projection** × **Vue** × **Modèle**) :

- La matrice **Modèle** qui modifie la position, la taille et l'orientation d'un maillage
- La matrice **Vue** qui définit les propriétés extrinsèques de la caméra (e.g. position, orientation)
- La matrice **Projection** qui contrôle les propriétés intrinsèques de la caméra (e.g. la focale).

Dire que les matrices **Vue** et **Projection** modifient la caméra n'est que la manière de penser de l'utilisateur mais pas ce qui se passe à l'intérieur d'OpenGL. En réalité, ce n'est pas la caméra qui change mais le monde autour d'elle qui se modifie pour exprimer les propriétés extrinsèques et intrinsèques de la caméra. Pour simplifier : "ce n'est pas la caméra qui avance, c'est le monde qui recule". Le produit de ces matrices permet de changer le repère dans lequel les coordonnées des sommets du maillage sont exprimées : du repère local vers le repère global, puis vers le repère caméra, et enfin dans le repère champ de vue (*frustrum*).

La matrice **MVP** est appliquée à la fin du *Geometry Shader* car après l'exécution de celui-ci, la géométrie n'est plus modifiée, et cette transformation doit toujours être la dernière. Mais il est également possible d'appliquer cette matrice dans le *Tessellation Evaluation Shader*, et même dans le *Vertex Shader* s'il n'y a pas d'étape de tesselation. L'application de la matrice **MVP** dans le *Geometry Shader* permet aussi de générer plusieurs objets identiques mais à des endroits différents en appliquant plusieurs matrices **MVP** distinctes sur la même primitive, et en ayant

en sortie une primitive par matrice. Par exemple, lors du rendu d'une armée, il est possible de ne calculer que la géométrie d'un unique soldat puis de l'afficher en plusieurs fois grâce au *Geometry Shader*.

### Le *Fragment Shader*

Pour chaque primitives qui sort du *Geometry Shader*, il y a une étape de rastérisation. Cette étape va aplatis l'espace sur un plan image par une projection orthogonale (la perspective est gérée par la matrice **Projection**), puis découper dans ce plan image le rendu qui sera afficher à l'écran (en supprimant tous les triangles qui ne sont pas visibles). Il ne reste alors plus qu'à colorer les triangles restants dans le *Fragment Shader*.

Le *Fragment Shader* reçoit un fragment en entrée et renvoie un fragment en sortie qu'il est chargé de colorer. La coloration de fragment est une partie très importante du rendu mais elle n'a que très peu d'intérêt du point de vue de la géométrie et seules les méthodes de colorations les plus simples (*e.g.* application de textures, rendu de type Blinn-Phong [Bli77]) ou montrant des informations géométriques (*e.g.* coloration du fragment en fonction de la normale) ont été implémentées. Une liste, loin d'être exhaustive, des utilisations courantes du *Fragment Shader* serait : rendu par lancer de rayon, gestion de la transparence, effets de particules (*e.g.* le brouillard), rendu des ombres par *shadow mapping* [Wil78], et même parfois la combinaison de plusieurs méthodes en exploitant la technique du rendu multiple.

### 5.1.2 Rendu de surfaces générées par un automate CIFS barycentrique

Il a été présenté dans les chapitres précédents comment calculer une tessellation de la surface limite d'un automate CIFS barycentrique. Dans cette sous-section, est décrit comment exploiter le *pipeline OpenGL* pour faire le rendu de ces tessellations. Comme leur nom le laisse penser, c'est l'étape de tessellation qui est au centre de ce rendu et c'est par celle-ci que commence la présentation. Les autres *shaders* seront ensuite présentés. Il est à nouveau conseillé de se reporter au schéma en [Figure 5.1](#) au cours de la lecture pour ne pas perdre le fil.

#### Reconstruction du patch de contrôle et choix du niveau de détails dans le *Tessellation Control Shader*

Le *Tessellation Control Shader* commence par reconstruire le patch de contrôle en cours de rendu. Parmi les types de faces reconnus par OpenGL, il y a le type appelé "patch" contenant un nombre quelconque de sommets mais comme ce nombre de sommets se doit d'être constant au cours du rendu, le type patch peut être utilisé pour les patches B-Splines ou NURBS qui contiennent toujours le même nombre de sommets mais pas pour les patches des surfaces de subdivision qui contiennent des sommets ou des faces extraordinaires. Pour passer outre cette restriction, trois *buffers* seront utilisés au lieu des deux habituellement requis :

- Le premier est le **buffer d'indexation des patches** qui est celui qui n'existe pas dans les implémentations habituelles. Celui-ci contient pour chaque patch : un pointeur sur le début d'un patch et la longueur de celui-ci.
- Le second est le **buffer des patches** qui contient tous les sommets indexés (un pointeur sur un sommet) des patches écrit les uns à la suite des autres.
- Le troisième est le **buffer des sommets modifiés** qui contient les coordonnées des sommets du maillage de contrôle après l'étape d'animation du *Vertex Shader*.

En suivant les pointeurs dans ces trois *buffers*, il est facile de reconstituer un patch de taille quelconque dans le *Tessellation Control Shader*, qui sera ensuite envoyé aux différents *Tessellation Evaluation Shader*. Comme le patch de contrôle n'a besoin d'être reconstruit qu'une seule fois, seule la première copie du *Tessellation Control Shader* (`gl_InvocationID = 0`) fait cette étape de reconstruction.

Ensuite, le *Tessellation Control Shader* reçoit du *Vertex Shader* un **représentant** du morceau de surface à afficher. Si le schéma est primal, le représentant est simplement la face centrale du patch. Si le schéma est dual, le représentant est construit en reliant l'ensemble des centres de gravité des faces incidentes au sommet central. Le *Tessellation Control Shader* calcule la projection de ce représentant sur le plan image (en appliquant la matrice **MVP**) et agit en fonction du résultat. S'il est en dehors du champ de vision, l'affichage lui est refusé en mettant à 0 son niveau de tessellation. S'il est visible, chacune de ses arêtes est subdivisée (`gl_TessLevelOuter[i]`) selon sa longueur (plus une arête est longue à l'écran, plus son niveau de détails doit être grand) par un nombre qui génère des sommets de l'*abstract patch* d'adresse finie pour le schéma. Les faces de l'*abstract patch* sont ensuite subdivisées

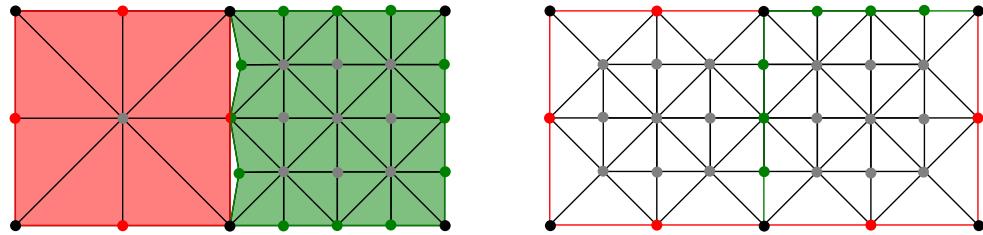


FIGURE 5.3 – Comparaison entre une tessellation par face et par arête. La tessellation par face (à gauche) crée une incohérence de topologie au niveau des raccords entre les faces ce qui peut entraîner l'apparition de *cracks*. A l'inverse, une tessellation par arête (à droite), construit forcément une topologie cohérente vu que l'arête est subdivisée de la même manière des deux côtés. Les sommets initiaux sont en noir, ceux ajoutés par subdivision d'arête sont en rouge (1 subdivision) et en vert (deux subdivisions), et ceux ajoutés par subdivision de face en gris.

(`gl_TessLevelInner[i]`) comme la valeur maximale des subdivisions d'arêtes. La subdivision en commençant par les arêtes permet d'éviter l'apparition de *cracks* (voir Figure 5.3).

### Calcul de la surface limite dans le *Tessellation Evaluation Shader*

Le *Tessellation Evaluation Shader* reçoit du *Tessellation Control Shader* le patch de contrôle et du *Tessellation Primitive Generator* les coordonnées de texture ( $u, v, w$ ) correspondant au sommet de l'*abstract patch* à calculer. Ces coordonnées de textures correspondent à un paramétrage du morceau de surface à afficher et donc à une adresse précise. Pour s'assurer que la tessellation soit cohérente, les paramètres renseignés dans le *Tessellation Control Shader* sont choisis de sorte que l'*abstract patch* ne contienne que des sommets d'adresse de longueur finie répartis uniformément sur l'espace des paramètres.

Le *Tessellation Evaluation Shader* doit calculer le sommet correspondant à ces coordonnées de texture sur le morceau de surface limite en cours de rendu. Pour cela, il doit trouver la combinaison barycentrique à appliquer sur le patch de contrôle. Le nombre de sommets du patch et les coordonnées de texture du sommet en cours de calcul permettent de retrouver l'indice de la combinaison barycentrique et sa longueur (le nombre de sommets du patch). Ensuite, il ne reste plus qu'à appliquer la combinaison sur le patch pour calculer le sommet du morceau de surface limite.

### Utilisation du *Vertex Shader* pour l'animation

Le *Vertex Shader* est souvent utilisé pour l'animation car il reçoit un sommet et en renvoie un autre. Dans le cas présent, l'animation se fait sur le maillage de contrôle, ce qui permet d'économiser beaucoup de temps de calcul. En effet, même si le résultat n'est pas tout à fait le même, l'animation du maillage de contrôle puis la subdivision est beaucoup plus rapide que l'animation d'un maillage déjà subdivisé. Avant de renvoyer un sommet, le *Vertex Shader* va l'écrire dans le **buffer des sommets modifiés** qui sera utilisé dans le *Tessellation Evaluation Shader* pour reconstituer le patch courant. Le sommet est tout de même envoyé au *Tessellation Control Shader* pour le choix du niveau de tessellation.

### Le *trimming* dans le *Geometry Shader*

Le *trimming* consiste à découper une partie d'une surface en faisant une restriction dans l'espace paramétrique. L'étape de *trimming* doit avoir lieu soit pendant la tessellation, soit après. Dans le cas présent, il a été décidé que cette étape aurait lieu dans le *Geometry Shader*. Celui-ci reçoit la restriction définie par des coordonnées locales de textures. Pour chaque primitive (triangle) de l'*abstract patch* reçu par le *Geometry Shader*, celui-ci teste pour chacun des trois sommets s'il est conservé ou dans la zone de restriction. Selon le nombre de sommets conservés, le *Geometry Shader* envoie une, deux ou aucune primitives au *Fragment Shader*. Les différents cas sont montrés dans la Figure 5.4.

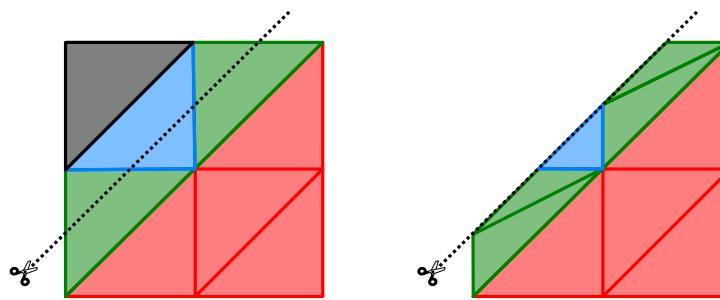


FIGURE 5.4 – Exemple des différents cas de *trimming* par le *Geometry Shader*. Les triangles dont 3 sommets sont conservés (en rouge) sont affichés, ceux à 2 sommets conservés (en vert) deviennent deux triangles, ceux à 1 sommet conservé (en cyan) sont modifiés et ceux dont aucun sommet n'est conservé sont supprimés (en noir).

- Si les 3 sommets sont conservés, la primitive est envoyée telle qu'elle est reçue
- Si 2 sommets sont conservés, un sommet est ajouté à chaque intersection d'une arête avec le bord extérieur de la restriction et le quadrangle obtenu est découpé en deux triangles
- Si 1 seul sommet est conservé, les deux sommets dans la restriction sont déplacés le long de l'arête jusqu'à atteindre le bord de la restriction
- Si aucun sommet n'est conservé, aucune primitive n'est envoyée au *Fragment Shader*

Le calcul de la projection du point d'intersection sur la surface limite peut être fait de deux manières différentes. La première consiste à le calculer par l'automate CIFS : d'abord l'adresse est générée, puis elle est déroulée sur l'automate, et enfin la combinaison calculée est appliquée sur le patch. Le problème c'est que cette méthode est lourde à mettre en place : le *Geometry Shader* doit reconstruire le patch, il doit également connaître l'algorithme de génération des adresses de l'automate CIFS et ses transformations. Même en supposant que tous ces points ont été résolus, il n'y a aucune assurance que l'adresse obtenue ne soit pas infinie et donc le résultat approché. La seconde méthode consiste simplement à définir le point d'intersection dans l'espace des paramètres comme une combinaison barycentrique des sommets de l'arête et d'appliquer cette combinaison sur les points de la surface limite correspondant à chaque sommet. Le point obtenu est une approximation du point réel mais dans le cas d'un morceau de surface suffisamment tesselé, l'erreur est minime. Des exemples de résultats obtenus par approximation barycentrique sont donnés en [Figure 5.5](#).

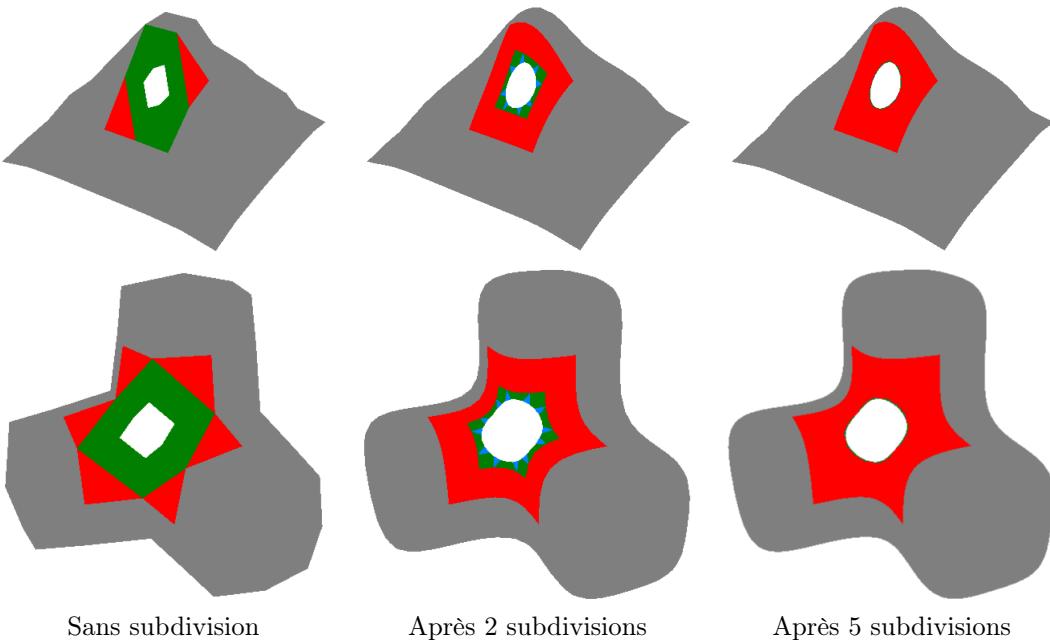


FIGURE 5.5 – Quelques exemples de *trimming* obtenu par approximation barycentrique à différents niveaux de subdivision. Les zones en gris ne sont pas concernées par le *trimming*. Les triangles des zones de *trimming* sont colorés de la même couleur que dans la [Figure 5.4](#).

Attention, deux cas de figure ne peuvent être gérés de cette manière : si la restriction est entièrement inclue dans une primitive OpenGL (dans le cas présent des triangles, mais la remarque est également valable pour les autres primitives *e.g.* quadrangles, lignes) car elle ne crée aucune intersection ou si la primitive fait plusieurs intersections avec la même arête. La plupart du temps, ce genre de problème peut être réglé en augmentant le niveau de tessellation du morceau de surface concerné.

### Le calcul des normales de la tessellation de la surface limite

Les normales de chaque sommet de la surface limite peuvent être calculées de deux manières différentes avant d'être utilisées pour le rendu dans le *Fragment Shader*. La première, la plus simple, consiste à calculer la normale de chaque primitive du *Geometry Shader*. Cette méthode est efficace quand la tessellation est très fine car les points de la primitive sont suffisamment proches pour que la normale de la primitive soit celle de la surface limite.

La seconde méthode, qui a l'avantage de générer les vraies normales de la surface limite, consiste à calculer ces normales dans le *Tessellation Evaluation Shader*. En même temps que l'application de la combinaison barycentrique permettant de trouver la position du sommet de la surface limite, deux autres combinaisons sont appliquées pour calculer les deux tangentes principales en ce sommet. Ensuite, il suffit de faire le produit-vectoriel de ces deux tangentes pour obtenir la normale. Pour placer la normale dans le repère de la caméra, il faut faire un calcul supplémentaire dans le *Geometry Shader* : la normale est transformée par la matrice  ${}^T(\mathbf{View} \times \mathbf{Model})^{-1}$ . La normale ne peut pas être calculée directement sur l'attracteur de l'espace barycentrique puis projetée dans l'espace géométrique car la projection conserve les tangentes mais pas les normales.

Le choix entre les deux méthodes dépend du niveau de tessellation de la surface limite : si elle est faible, la seconde méthode est à privilégier pour ne pas obtenir de décrochement de normale alors que si elle est élevée, la première méthode permet un résultat suffisamment proche de la réalité avec des calculs beaucoup plus rapides. Des comparaisons entre les deux méthodes sont données en Figure 5.6.

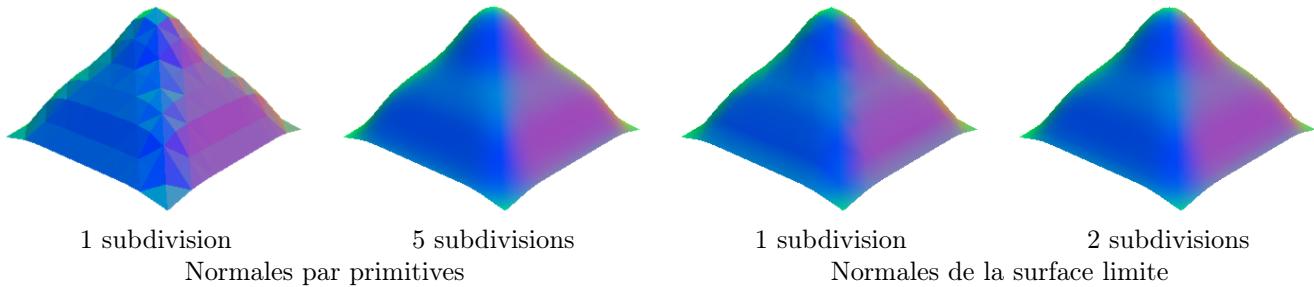


FIGURE 5.6 – Comparaison entre les deux méthodes de calcul des normales pour différents niveaux de tessellation. Il faut noter que pour la même qualité de normales, seules deux subdivisions sont nécessaires en calculant la normale de la surface limite alors que cinq sont requises dans le calcul par primitive. Les normales sont colorées en rouge si elles pointent vers la droite de l'image, en vert vers le haut, et en bleu vers la caméra, avec des mélanges possibles. Plus les changements de normales entre deux faces sont doux, plus les dégradés de couleurs sont lissés.

### 5.1.3 Construction des différents automates CIFS pour le remplissage des buffers

Jusqu'à présent, seul le *pipeline* a été présenté mais pourtant celui-ci va chercher des informations dans les *buffers*. Le remplissage de ces différents *buffers* avant le rendu est présenté dans cette sous-section. Comme le remplissage est différent selon l'automate CIFS à appliquer, une classe **CIFS** a été construite comme un conteneur des classes abstraites suivantes :

- **Topologie\_patches** qui transforme un maillage de type sommet-face en une liste de patches et les écrit dans les trois *buffers* contenant les patches et les sommets indexés.
- **Paramétrisation** qui génère les différentes adresses de l'automate et sélectionne le *Tessellation Control Shader* et le *Tessellation Evaluation Shader* à utiliser dans le *pipeline*.
- **Combinaisons\_barycentriques** qui calcule les combinaisons et les écrit dans le *buffer* correspondant.

L'implémentation est pensée de manière modulaire : pour construire une classe CIFS chacune des classes abstraites doit être surchargée. L'intérêt vient du fait que plusieurs automates partagent certaines de ces classes, mais qu'elles ne doivent être écrites qu'une seule fois.

### Construction des patches à partir du maillage de contrôle

Le maillage de contrôle est représenté de manière classique : par une liste de sommets (où chaque sommet possède 3 coordonnées) et par une liste de faces (qui contient l'index des sommets à utiliser). La liste des sommets est inchangée, elle est transcrise à l'identique dans le **buffer des sommets modifiés** et également envoyée au *Vertex Shader*.

Ensuite les patches sont construits à partir de la liste des faces. Cette étape est très fastidieuse à implémenter car il faut s'assurer qu'aucun patch ne soit oublié dans la construction et que l'organisation soit constante et cohérente dans la totalité des patches construits. Quelques exemples de topologies irrégulières de patches sont donnés en Figure 5.7.

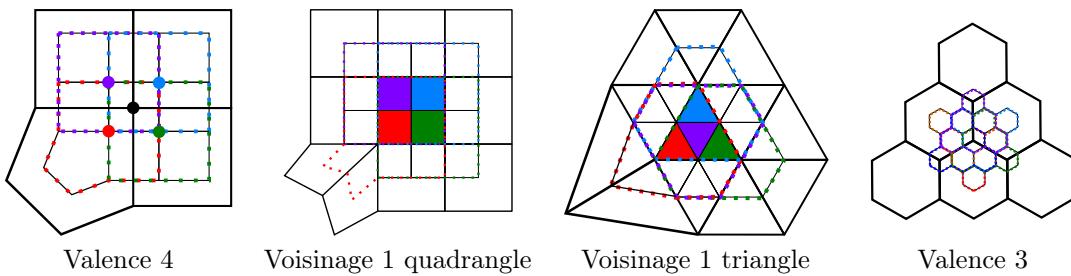


FIGURE 5.7 – Différents exemples de topologie irrégulière de patches. La topologie "Valence 4" concerne les schémas de Doo-Sabin, *Mid-edge*, et les surfaces tensorielles B-Splines quadratiques (dans le cas régulier) ; le "Voisinage 1 quadrangle" le schéma de Catmull-Clark et les surfaces tensorielles B-Splines cubiques (dans le cas régulier) ; le "Voisinage 1 triangle" le schéma de Loop et les Box-Splines (dans le cas régulier) ; le "Valence 3" le schéma *Honeycomb*.

Dans le cas de l'affichage de surface de type NURBS, la classe **Topologie\_patches** est également chargée de l'uniformisation de chaque patch du maillage avant de les envoyer à la carte graphique. Les sommets sont alors dotés d'une coordonnée supplémentaire pour devenir des points massiques.

### Paramétrisation de la surface limite et génération des adresses

La paramétrisation de la surface limite dépend principalement de la forme du morceau de surface généré (triangle ou quadrangle) et du nombre de transformations de l'automate CIFS. Différents exemples sont présentés en Figure 5.8.

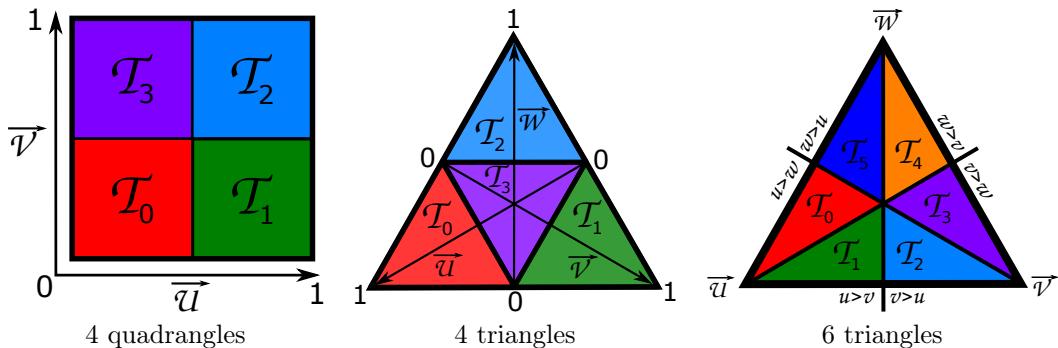


FIGURE 5.8 – Quelques exemples de paramétrisation de morceaux de surface limite.

La paramétrisation permet de construire une tessellation choisie de sorte à n'avoir que des sommets d'adresses finies répartis uniformément sur l'espace des textures. Pour cela, il suffit de générer toutes les adresses possibles à partir de la paramétrisation et de supprimer les doublons. Ces adresses sont ensuite envoyées à la classe **Combinaisons barycentriques** pour qu'elle calcule les combinaisons correspondant à chaque adresse et les écrivent dans le **buffer des combinaisons barycentriques** à un index donné.

Le *Tessellation Control Shader* de l'automate est implémenté de manière à donner au *Tessellation Primitive Generator* les informations entraînant la construction d'un *abstract patch* identique à cette tessellation. Le *Tessellation Evaluation Shader* doit lire les coordonnées de texture d'un sommet de cet *abstract patch* et en déduire l'adresse du sommet et donc l'index de la combinaison barycentrique à appliquer sur le patch.

### Calcul des combinaisons barycentriques

La classe qui génère les combinaisons barycentriques est unique pour chaque automate CIFS (contrairement aux deux classes présentées juste avant qui peuvent être communes à plusieurs schémas). Dans cette classe sont écrites les matrices  $\mathcal{M}_0(k) \dots \mathcal{M}_{n-1}(k)$  en fonction de la valence  $k$  (la liste des valences possibles est stockée dans la classe). Les transformations points-fixes  $\mathcal{P}_i$  sont calculées par analyse des vecteurs-propres par une bibliothèque externe d'analyse (e.g. Armadillo, Eigen).

Pour une adresse reçue de la classe **Paramétrisation**, la combinaison barycentrique de chaque valence de la liste est générée. Une fois les combinaisons générées, elles sont rangées dans le **buffer des combinaisons barycentriques** à l'index reçu de la classe **Paramétrisation**.

#### 5.1.4 Comparaison avec les méthodes classiques de rendu temps réel

Dans un premier temps seront présentées les deux méthodes classiques de rendu des surfaces de subdivision en temps réel. Ces méthodes sont considérées comme des méthodes de subdivision mais elles sont valables pour toutes les constructions par itérations uniformes et stationnaires (e.g. les fractales). Ensuite la méthode des *shaders* de combinaisons barycentriques est comparée aux deux autres sur deux points : la mémoire consommée et le temps de calcul, dans deux cas différents : avec animation par squelette et sans animation. Pour simplifier les comparaisons et rester dans le sujet de la modélisation géométrique, seules les informations et les calculs permettant de générer les primitives avant le *Fragment Shader* seront pris en compte. Tout ce qui concerne le rendu (e.g. coordonnées de textures, normales) est simplement ignoré ; comme si le *Fragment Shader* se contentait d'afficher des triangles de couleur constante dans une lumière ambiante fixe. Comme les comparaisons dépendent du schéma choisi, elles seront toutes estimées pour le schéma de Catmull-Clark qui est le plus courant.

##### Les méthodes classiques de rendu temps réel des surfaces de subdivision.

En 40 années d'existence, il y a eu énormément d'implémentations différentes pour le rendu temps réel des surfaces de subdivision. Ici seront présentées les plus courantes mais qui ne sont probablement pas les plus efficaces.

La première est la plus simple, il s'agit de la **subdivision préalable**. Les règles de subdivision sont implémentées dans la partie processeur du programme et les tessellations obtenues pour différents niveaux d'itérations sont stockées comme plusieurs maillages distincts. Lors du rendu, une première étape de sélection du niveau de détails est effectuée et le maillage choisi est envoyé à la carte graphique pour le rendu. Cette méthode consomme beaucoup de mémoire car pour un rendu très fin, un maillage très lourd est envoyé à la carte graphique. De plus, elle ne permet pas d'avoir différents niveaux de détails au sein du même maillage.

La seconde méthode est apparue avec l'apparition des *Geometry Shader* et a été déplacée par la suite sur l'étape de tessellation plus récente. Cette méthode de **subdivision multi-passes** est très économique en mémoire car seuls les patches du maillage de contrôle sont envoyés à la carte graphique mais nécessite beaucoup de temps de rendu car les différentes étapes de subdivision sont appliquées directement sur la carte graphique en plusieurs passes.

La méthode des *shaders* de combinaisons barycentriques est un compromis entre ces deux méthodes. Elle peut d'ailleurs être considérée en quelque sorte comme une méthode hybride des deux autres (le seul maillage reçu par la carte graphique est le plus grossier mais des informations nécessaires aux différents niveaux de tessellation sont également envoyées). Avant de parler de mémoire ou de temps de calcul, le premier avantage de la méthode des

*shaders* de combinaisons barycentriques sur les deux autres est qu'elle sépare les différentes parties (topologie, géométrie, et paramétrisation) qui sont habituellement mélangées dans les règles de subdivision ce qui permet une implémentation beaucoup plus modulaire que les autres.

### La mémoire

La mémoire est composée de trois types d'informations : les sommets, la topologie (les faces ou les patches), et les combinaisons barycentriques (uniquement dans la méthode des *shaders* de combinaisons barycentriques).

Un **sommet** contient ses coordonnées (3 floats) et dans le cas de l'animation un index (1 unsigned int) de matrice et un coefficient d'influence (1 float) pour chaque os (par exemple il y a 4 os par sommet sur Unity). Lorsqu'un sommet du maillage est influencé par un ou plusieurs os, il est transformé en lui appliquant indépendamment la matrice de chaque os et les différents résultats sont ensuite fusionnés selon l'influence que chaque os a sur le sommet. Donc un sommet pèse 12 octets et un sommet animé 44 octets. Le nombre de sommets du maillage de contrôle est  $S_0$  et celui d'un niveau d'itération  $i$  est  $S_i \approx 4^i S_0$  (pour le schéma de Catmull-Clark).

La **topologie** dépend du schéma mais dans le cas de Catmull-Clark le nombre de sommets par face est 4 (pour la subdivision préalable) et le nombre de sommets par patch (pour les méthodes où la subdivision est faite sur la carte graphique) est la plupart du temps 16 (car la plupart des patches sont réguliers) avec le patch qui est lui même indexé par un pointeur et une longueur (2 unsigned int). Les sommets sont indexés par des pointeurs (1 unsigned int) et donc le poids d'une face est 16 octets et celui d'un patch est 72 octets. Le nombre de face est  $F_0$  pour le maillage original et  $F_i = 4^i F_0$  pour le niveau d'itération  $i$ .

Le poids des **combinaisons barycentriques** dépend de la taille des combinaisons, du nombre de valences différentes et du niveau de tessellation maximum. Pour le schéma de Catmull-Clark, si les valences de 3 à 10 sont considérées, la taille d'une combinaison est en moyenne de 21 coefficients (floats). Le nombre de valences différentes est 8. En considérant un niveau de tessellation maximal  $\mathcal{I}$ , il y a  $(2^{\mathcal{I}} + 1)^2$  sommets dans l'*abstract patch* (et donc de combinaisons).

Le nombre d'éléments de chaque type est résumé dans le tableau suivant :

Nombre de ... (poids)	Subdivision préalable	Subdivision multi-passes	Combinaisons barycentriques
Sommets (12o ou 44o)	$4^i * S_0$	$S_0$	$S_0$
Face (16o) / patches (72o)	$4^i * F_0$	$F_0$	$F_0$
Coefficients (4o)	0	0	$21 * 8 * (2^{\mathcal{I}} + 1)^2$

Pour information la mémoire consommée qui est identique pour l'ensemble des méthodes et est dérisoire par rapport au poids du maillage (donc ignorée dans le calcul) est :

- La matrice **MVP** du modèle (une matrice 4x4 de float donc 64 octets)
- La matrice d'animation pour chaque os (64 octets par os)

En posant  $S_0 = F_0 = 1000$  et  $\mathcal{I} = 5$  le tableau suivant est obtenu :

Itérations	Sans animation			Animation par squelette		
	Préalable	Multi-passes	Combinaisons	Préalable	Multi-passes	Combinaisons
i = 0	28 ko	84 ko	816 ko	60 ko	116 ko	848 ko
i = 1	112 ko	84 ko	816 ko	240 ko	116 ko	848 ko
i = 2	448 ko	84 ko	816 ko	960 ko	116 ko	848 ko
i = 3	1,8 Mo	84 ko	816 ko	3,8 Mo	116 ko	848 ko
i = 4	7,1 Mo	84 ko	816 ko	15 Mo	116 ko	848 ko
i = 5	28 Mo	84 ko	816 ko	61 Mo	116 ko	848 ko

La subdivision préalable n'est intéressante que dans le cas où il n'y a pas de subdivision car dans ce cas le même nombre de sommets est envoyé mais la topologie des patches est plus coûteuse que celle des faces. Mais dès que le maillage est subdivisé une fois, la subdivision multi-passes est plus économique en mémoire. Il faut attendre la troisième itération pour un maillage fixe et la seconde pour un maillage animé pour que la méthode des combinaisons barycentrique devienne plus économique que celle de la subdivision préalable. La méthode multi-passes est toujours plus légère que celle des combinaisons barycentriques car les deux méthodes envoient les mêmes données pour le maillage mais celle des combinaisons envoie en plus les combinaisons barycentriques. Par contre, il faut préciser que cette partie n'est envoyée qu'une seule fois à la carte graphique : il n'est pas nécessaire de la renvoyer pour chaque maillage. Ainsi, le coût en mémoire affiché dans le tableau n'est valable que pour le premier maillage, tous les autres ont le même coût que pour la méthode de multi-passes.

### Temps de calcul

Les étapes du *pipeline* (avant l'étape de rendu) sont : animation par squelette des sommets, sélection du niveau de tessellation, et génération des primitives.

Lors de l'**animation par squelette** d'un sommet, une matrice de transformation  $4 \times 4$  par os associé (en moyenne 2) à ce sommet est appliquée. Les différents sommets obtenus sont ensuite interpolés selon les coefficients associés à chaque os. Le coût total de l'animation est la somme de la recherche des matrices (2 indexations), de l'application de celles-ci (2 fois 16 produits et 9 sommes) et le mélange des sommets obtenus (2 produits et 1 somme) ; chaque sommet coûte donc 2 indexations + 34 produits + 19 sommes.

La **sélection du niveau de tessellation** n'a pas lieu dans le *pipeline* pour les subdivisions préalables. Pour les autres, il peut être fait dynamiquement dans le *Tessellation Control Shader* en projetant les 4 arêtes de la face ou du représentant du morceau de surface limite sur le plan image par l'application de la matrice **MVP** (4 fois 16 produits et 9 sommes) et en sélectionnant le niveau de tessellation pour chaque arête (4 indexations) et pour la face (choisie en même temps que les arêtes). Le coût global de cette sélection est donc de 4 indexations + 64 produits + 36 sommes pour chaque face du maillage de contrôle  $F_0$ .

Dans les deux méthodes de génération de la subdivision directement sur la carte graphique, il faut commencer par une **reconstruction du patch** à calculer ce qui nécessite 1 indexation de patch et 16 indexations de sommets par patches (faces) du maillage de contrôle.

La **génération des primitives** est soit l'application des règles de subdivision dans la méthode de subdivision à la volée, soit l'application des combinaisons barycentriques dans la méthode des *shaders* de combinaisons barycentriques.

Pour le schéma de Catmull-Clark, chaque étape de subdivision consiste à la création d'un sommet par face (4 produits et 3 sommes), par arête (6 produits et 5 sommes), et par sommet (9 produits et 8 sommes dans le cas d'un patch régulier). Un maillage est généralement composé d'autant de sommets que de faces et de deux fois plus d'arêtes. Ainsi, le passage du niveau d'itération  $i$  au niveau  $i + 1$  nécessite 25 produits et 21 sommes pour chaque face (au nombre de  $F_i$ ). Le coût global de l'opération est la somme des coûts de passage entre le niveau de tessellation 0 (le maillage de contrôle) et le niveau  $i$ .

Pour l'application de la combinaison barycentrique, il faut considérer le coût de la recherche de la combinaison barycentrique (1 indexation). Ensuite celle-ci est appliquée sur le patch reconstitué (16 produits et 15 sommes dans le cas d'un patch régulier). Pour chaque sommet du niveau de tessellation  $i$  ( $S_i \approx 4^i S_0$ ) le coût de calcul est de 1 indexation + 16 produits + 15 sommes auquel il faut ajouter le coût de reconstruction du patch.

Les opérations d'indexation, de somme et de multiplication ne demandent pas toutes le même temps de calcul. Mais comme ce temps de calcul dépend des architectures de cartes graphiques, qui en plus est amené à évoluer, les trois opérations seront toutes considérées comme **opérations de base**. Le nombre d'opérations de base pour chaque méthode et chaque niveau de tessellation est résumé dans le tableau suivant (pour  $S_0 = F_0 = 1000$  et  $\mathcal{I} = 5$ ) :

Méthodes	Calcul	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5
Préalable	Animation	55 k	220 k	880 k	3,5 M	14 M	56 M
	Total	55 k	220 k	880 k	3,5 M	14 M	56 M
Multi-passes	Animation	55 k					
	Sélection	104 k					
	Reconstruction	18 k					
	Génération	0	184 k	920 k	3,8 M	16 M	63 M
	Total	177 k	361 k	1 M	4 M	16 M	63 M
Combinaisons	Animation	55 k					
	Sélection	104 k					
	Reconstruction	18 k					
	Génération	35 k	140 k	560 k	2,2 M	9 M	36 M
	Total	212 k	317 k	737 k	2,4 M	9,1 M	36 M

Dans le cas d'un maillage fixe, la subdivision préalable est incontestablement la méthode la plus efficace car son temps de calcul (hors temps d'affichage) est nul. A l'inverse, dans le cas d'un maillage animé, elle est dépassée par les deux autres dès la deuxième itération. En comparant les deux autres méthodes, il apparaît que la méthode des combinaisons barycentriques est plus efficace en calcul et que cette efficacité augmente avec le nombre d'itérations.

### Résumé des comparaisons

Les différentes méthodes de génération de la géométrie des surfaces de subdivision sont comparées en [Figure 5.9](#). En résumé, la méthode de subdivision préalable est la plus intéressante pour les maillages inanimés sauf pour des maillages très lourds nécessitant un traitement spécial. Pour un maillage animé, la méthode de génération de la tessellation à la volée par combinaisons barycentriques est à privilégier car c'est la plus rapide en calcul sans pour autant être beaucoup plus lourde en mémoire que la méthode de la subdivision multi-passes.

### 5.1.5 Améliorations possibles

Le modèle présenté dans cette section a fait l'objet d'un logiciel appelé Subdiv entièrement implémenté pour les besoins de la thèse. Quelques exemples de résultats obtenus par le logiciel sont montrés en [Figure 5.10](#). Les temps de calcul nécessaires pour la tessellation et l'affichage en couleur uniforme de la surface limite du schéma de Catmull-Clark sur une tour Dell Precision T7600 (processeur : Intel Xeon CPU E5-2609 @ 2.40 GHz x8, carte graphique : Nvidia Quadro K2000/PCIe/SSE2) sont donnés dans le tableau suivant :

Maillage	Gaussienne		Tetris		Tête		Humain	
Sommets	100		74		4276		13 652	
Faces	81		72		4249		13 650	
Iteration	ms	FPS	ms	FPS	ms	FPS	ms	FPS
i = 0	0.21	4600	0.23	4400	0.79	1260	3.2	320
i = 1	0.22	4500	0.24	4300	1.25	800	3.8	260
i = 2	0.23	4200	0.25	4000	3.4	300	8	125
i = 3	0.37	2700	0.41	2400	16	60	32	30
i = 4	0.76	1300	1.02	975	62	16	125	8
i = 5	1.66	600	2.33	430	167	6	333	3

La première amélioration serait l'ajout des schémas de subdivision présentés dans cette thèse mais jamais implémentés. En particulier, les schémas à support non-compact qui nécessitent une réelle réflexion pour la gestion des différentes irrégularités possibles. De plus, l'implémentation de la méthode d'uniformisation des patches de NURBS permettrait l'affichage des surfaces NURBS tensorielles ce qui serait un vrai plus pour le logiciel.

Ensuite la gestion des schémas à topologies fractales lacunaires. Pour l'instant les lacunarités sont gérées par le *Geometry Shader* : soit par *trimming*, soit en refusant l'affichage de certaines primitives. Pour être conforme à la réalité mathématique, il faudrait que les lacunarités soient des conséquences de l'automate et donc que le *Tessellation Evaluation Shader* les génère mais le *pipeline OpenGL* ne permet pas d'agir sur la topologie de l'*abstract patch* et de supprimer des faces dès le *Tessellation Evaluation Shader*.

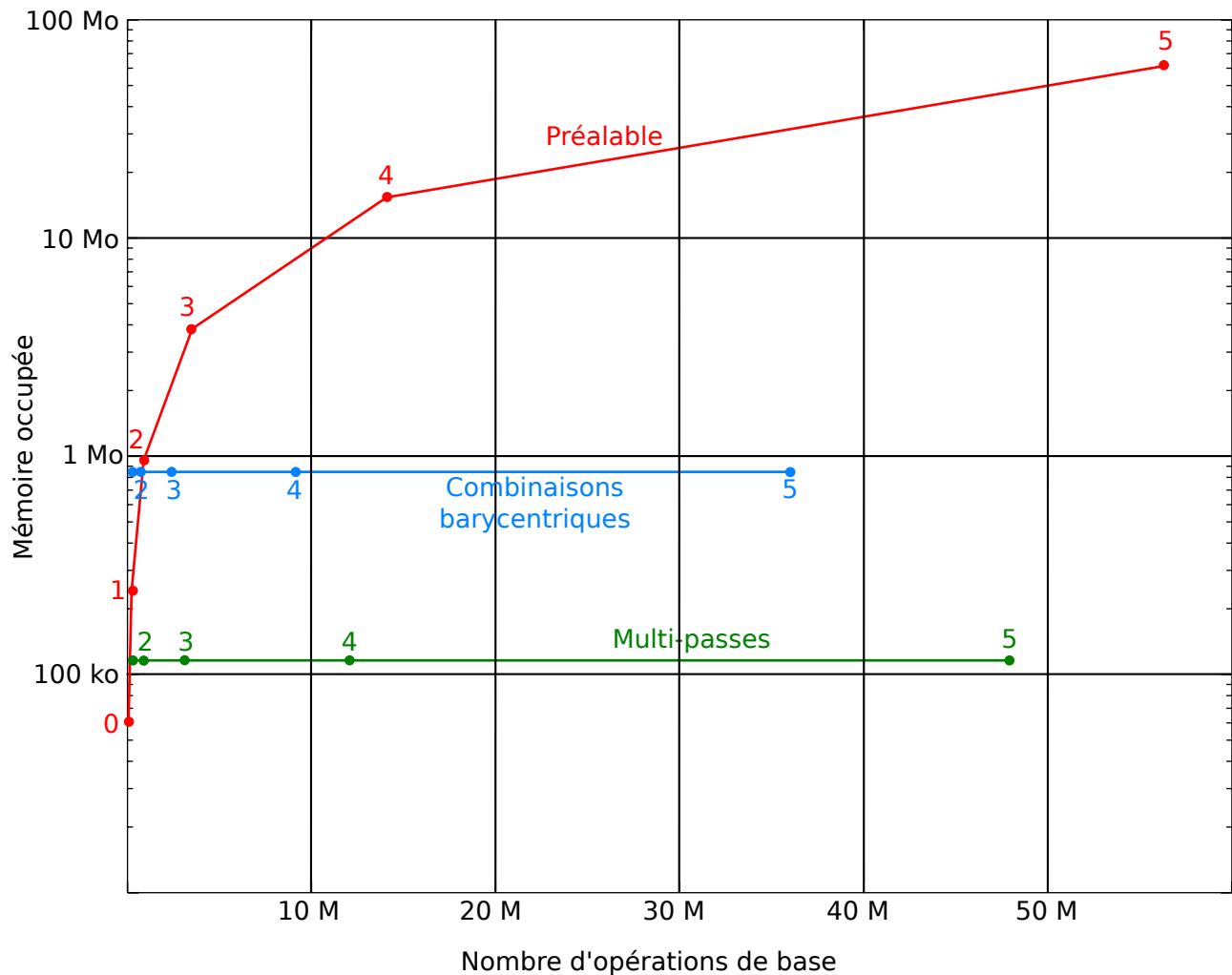


FIGURE 5.9 – Comparaison entre les trois méthodes de modélisation géométrique des surfaces de subdivision en fonction du temps de calcul et de la mémoire occupée sur un maillage animé de 1000 faces et 1000 sommets. Les numéros à côté des points indiquent le nombre d'itérations appliquées. Pour les courbes multi-passes et combinaisons barycentriques, les points 0 et 1 sont omis car beaucoup trop proches des points 2. La mémoire est représentée sur une échelle logarithmique à cause de l'explosion en mémoire de la courbe de subdivision préalable.

## 5.2 MODITERE : un logiciel d'optimisation topologique

MODITERE est un modeleur de CGAO développé par le Laboratoire d'Informatique de Bourgogne (anciennement LE2I : Laboratoire Electronique, Informatique et Image) dans le cadre du projet ANR MODITERE (n° ANR-09-COSI-014) et actuellement maintenu par Gilles Gouaty. Le logiciel utilise une représentation par automates CIFS barycentriques des courbes, des surfaces, et des volumes ainsi que les différents outils offerts par les CIFS pour une utilisation à but industriel et en particulier pour l'optimisation topologique.

### 5.2.1 Présentation du logiciel avant le début de la thèse

L'optimisation topologique consiste à concevoir une forme qui minimise la quantité de matière tout en satisfaisant les **contraintes** du système. Les contraintes, qui doivent absolument être résolues, peuvent être par exemple :

- contrainte de bord : la surface extérieure de l'objet doit être exactement celle donnée par l'utilisateur,
- contraintes mécaniques : résistance, flexibilité,
- contraintes de fabricabilité qui dépendent de la machine qui va construire l'objet (*e.g.* une imprimante 3D).

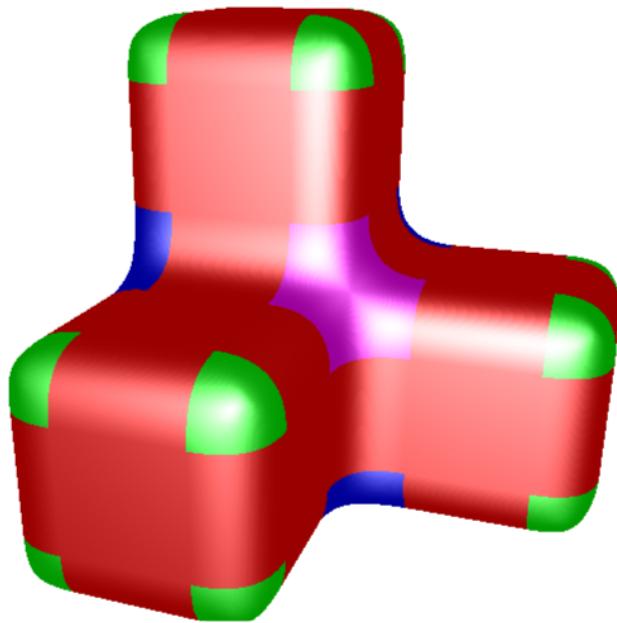


Schéma de Doo-Sabin utilisé sur le maillage Tétris

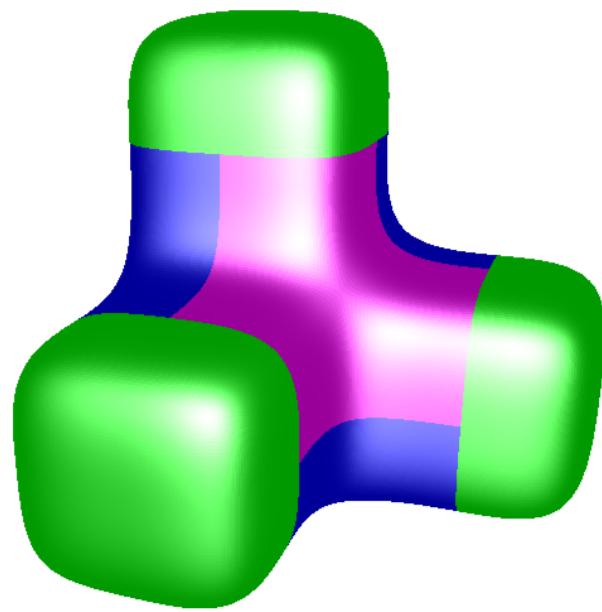


Schéma de Catmull-Clark sur le maillage Tétris

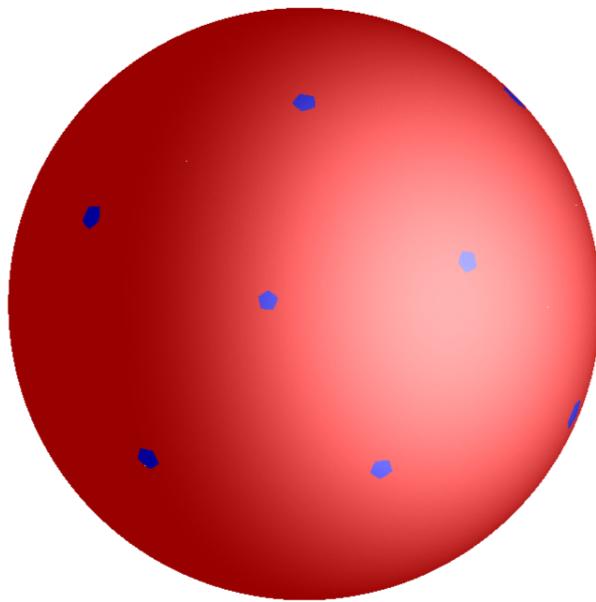


Schéma de Loop utilisé sur le maillage Icosphère

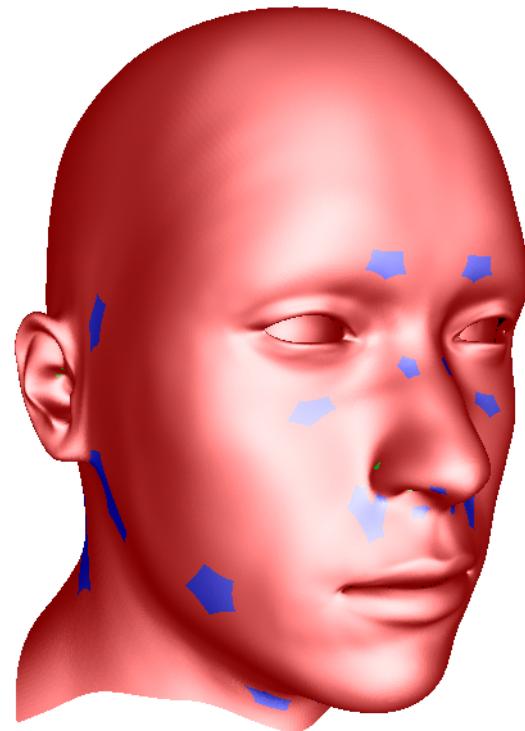


Schéma de Catmull-Clark utilisé sur le maillage Tête

FIGURE 5.10 – Différents exemples de rendu par l'utilisation de *shaders* de combinaisons barycentriques. Les morceaux de surface en rouge sont issus des patches réguliers, ceux en vert d'un patch irrégulier où la valence du sommet irrégulier est  $k = 3$  ou le nombre de sommet de la face irrégulière est  $s = 3$ . En bleu, les patches irréguliers  $k = 5$  ou  $s = 5$  et en magenta les patches  $k = 6$  ou  $s = 6$ .

Une fois les contraintes définies, il faut choisir une **fonction de coût** qui attribue une influence à chacun des objectifs à optimiser. Par exemple :

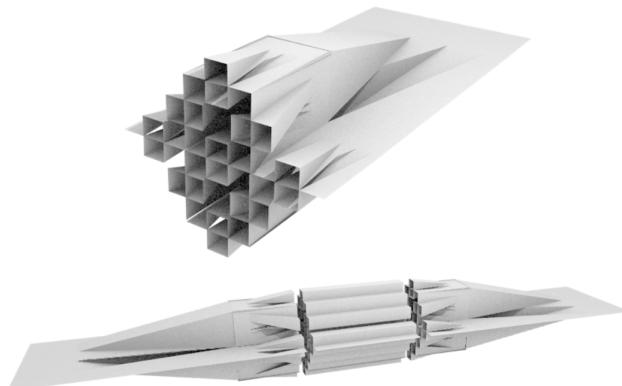
- optimisation de la quantité de matière utilisée (pour des raisons économiques ou de masse de l'objet),
- optimisation thermique pour une meilleure évacuation de la chaleur,
- optimisation de l'écoulement/de la rétention des fluides.

Les paramètres de construction de la surface et la fonction de coût sont ensuite envoyés à un **solveur** qui va agir sur les paramètres pour minimiser la fonction.

Voici deux exemples concrets où le logiciel a été utilisé. Le premier est un **support de presse de moulage par injection** : du plastique fondu est injecté sous pression dans un moule. La contrainte principale est la résistance à cette forte pression. L'utilisation d'une structure arborescente et de canaux de refroidissement limite l'inertie thermique du moule et ainsi réduit à la fois le temps de chauffe nécessaire à l'injection et le temps de refroidissement avant démoulage. Le résultat est visible en [Figure 5.11](#)



Supports de presse de moulage



Echangeur thermique

[FIGURE 5.11](#) – Deux exemples de réalisations faites sur le logiciel MODITERE.

Le second est la création d'un **échangeur thermique** dont les couches principales correspondent à différentes itérations de la courbe de Peano [\[Pea90\]](#) et les couches intermédiaires interpolent les deux couches principales qui les encadrent. Ainsi, lorsqu'un liquide à refroidir et un liquide de refroidissement sont injectés dans l'échangeur, ceux-ci sont de plus en plus mélangés au fur et à mesure qu'ils progressent dans les couches (voir [Figure 5.12](#)). La dernière couche est un tamis qui maximise l'aire de la surface de séparation des deux liquides ce qui a pour conséquence d'optimiser les échanges thermiques. Le résultat final est montré en [Figure 5.11](#).

### 5.2.2 Les apports de la thèse dans le modeleur MODITERE

Les travaux de cette thèse ont permis l'implémentation dans le logiciel de CGAO MODITERE de deux nouveaux types de surface : les surfaces de subdivision et les surfaces NURBS tensorielles. Les surfaces de subdivision n'étant que peu utilisées en CGAO, leur intégration dans MODITERE n'a pour l'instant servi dans aucune optimisation topologique à but d'industrialisation. Par contre, il sera présenté dans la sous-section suivante leur intérêt dans le processus global de CGAO et dans la résolution de la problématique de cette thèse.

Jusqu'à présent, le logiciel utilisait des structures fractales pour créer des supports arborescents ou des volumes lacunaires remplissant des pièces de CGAO représentées par des surfaces tensorielles de Bézier ou B-Spline de bas degré. Avec l'ajout des surfaces NURBS tensorielles de degré quelconque, il est désormais possible de gérer des objets représentés par des NURBS. Comme les NURBS sont le modèle de représentation le plus courant dans l'industrie, leur description sous forme d'automate CIFS est une avancée importante pour le logiciel et l'acceptation des CIFS comme modèle de CGAO à but industriel. Le logiciel permettant également de faire des raccords entre des surfaces de natures différentes, il peut dorénavant faire ces raccords avec des surfaces NURBS tensorielles. Des exemples de résultats obtenus par MODITERE sont donnés en [Figure 5.13](#).

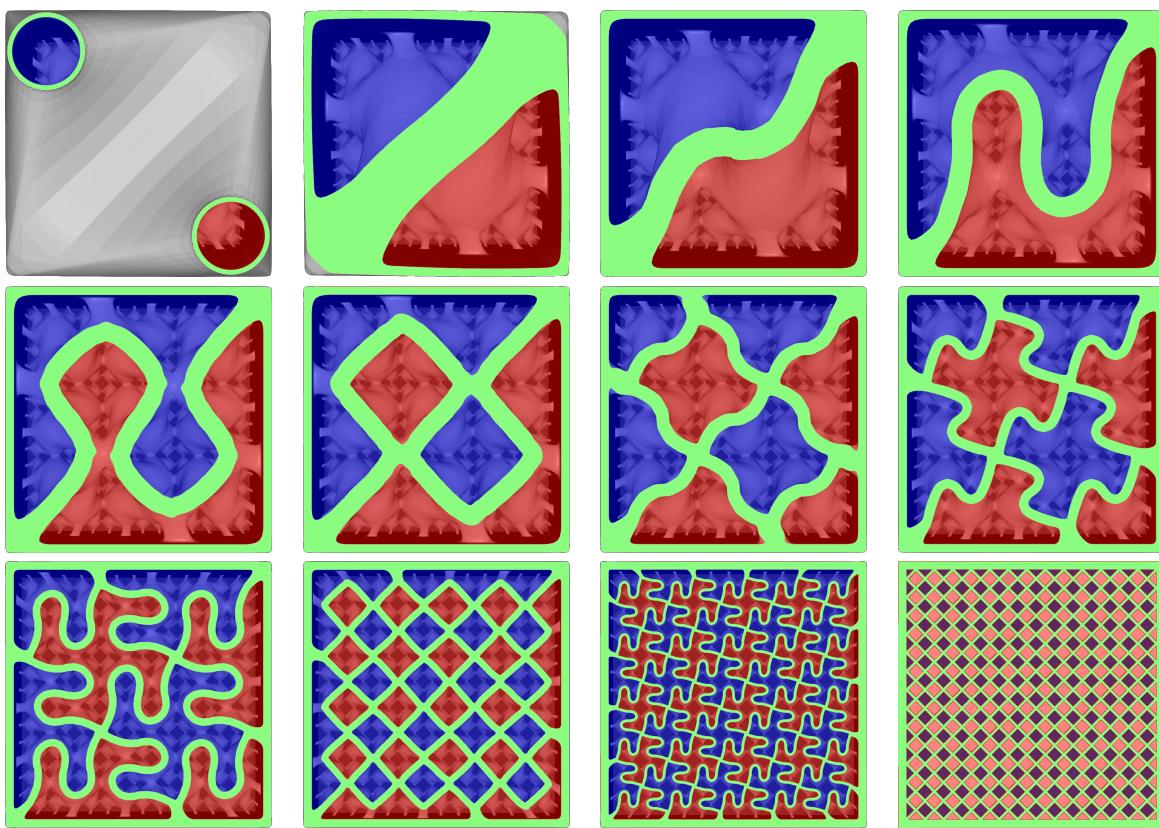


FIGURE 5.12 – Différentes coupes de l'échangeur thermique conçu sur le logiciel MODITERE. En rouge le liquide à refroidir, en bleu le liquide de refroidissement et en vert la surface de séparation des deux liquides où à lieu l'échange thermique.

### 5.2.3 Vers un modèle géométrique générique à base de CIFS

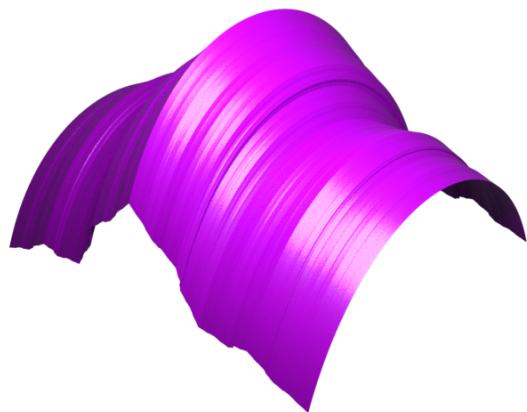
Avec l'intégration des NURBS et des surfaces de subdivision dans le logiciel MODITERE, et la possibilité de raccorder ces surfaces grâce à leur représentation sous forme d'automates CIFS, un début de résolution de la problématique se profile. Pour rappel, l'objectif de la thèse est de trouver une méthode permettant au *designer* qui utilise des surfaces de subdivision et à l'ingénieur qui a besoin des NURBS pour ses calculs de test de travailler sur le même modèle. Dans cette sous-section est présenté une ébauche de solution qui apporte en plus la possibilité de rajouter une étape d'optimisation topologique en fin de processus.

#### **Etape 1 : forme générale et esthétisme**

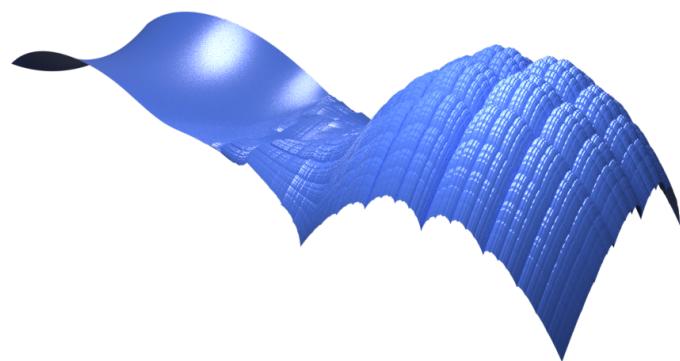
La première étape est celle de l'esquisse du *designer* donnant au modèle sa forme générale et son esthétisme. Celui-ci privilégie l'utilisation des surfaces de subdivision (et vraisemblablement de celles de Catmull-Clark) car elles sont instinctives d'utilisation et permettent une grande liberté artistique. Le modèle est ensuite envoyé à l'ingénieur.

#### **Etape 2 : ingénierie**

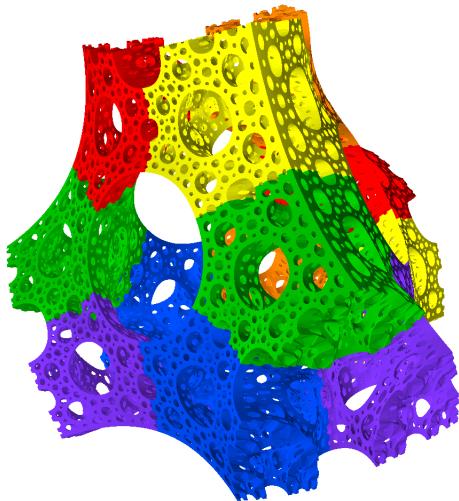
Dès que le modèle est reçu, toutes les zones sans sommet extraordinaire sont converties en maillage de surfaces NURBS tensorielles. Pour le schéma de Catmull-Clark, la conversion est parfaite et les surfaces NURBS obtenues sont bi-cubiques et tous les vecteurs nodaux sont uniformes. Les raccords entre les zones NURBS et les zones de surfaces de subdivision sont assurés par l'outil de raccord d'automates. Le modèle est alors composé de trois zones : les zones de subdivision définies par le *designer* qui sont fixées, les zones de raccord mises à jour automatiquement à chaque modification, et les zones NURBS sur lesquelles l'ingénieur va agir. L'ingénieur doit alors modifier les parties NURBS pour certaines optimisations (*e.g.* l'aérodynamisme de la carrosserie d'une voiture ou du fuselage d'un avion). Le modèle est ensuite renvoyé au *designer* pour validation.



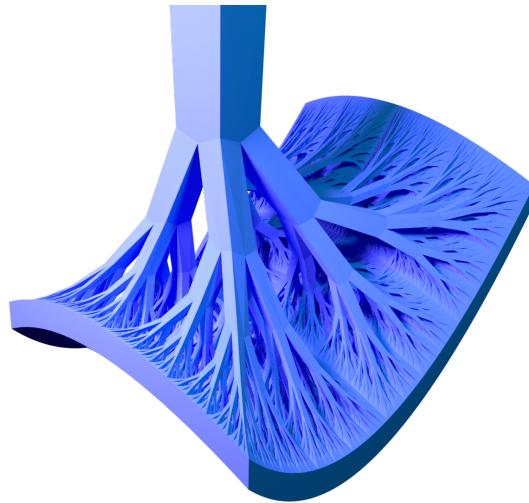
Surface tensorielle Takagi-NURBS



Raccord entre des surfaces fractale et NURBS



Lacunarités définies par des NURBS



Arbre dont les feuilles convergent vers une surface NURBS

FIGURE 5.13 – Quelques exemples de réalisations impliquant des NURBS sur le logiciel MODITERE du LIB : une surface tensorielle générée à partir d'une NURBS et d'une courbe de Takagi ; un raccord entre deux surfaces tensorielles, l'une NURBS et l'autre de Takagi ; un volume lacunaire dont les lacunarités sont décrites par des NURBS ; une structure arborescente dont les feuilles correspondent à des points d'une surface tensorielle NURBS.

### **Etape 3 : validation**

L'étape de validation est la plus compliquée car elle nécessite de trouver un compromis entre le *designer* et l'ingénieur. Dans les processus habituels, cette étape est très fastidieuse car chaque modification sur le modèle fait par l'un nécessite une conversion du modèle pour la répercuter chez l'autre. Et cette conversion est généralement faite par une tierce personne qui malgré une assistance automatisée doit faire énormément de travail à la main.

Dans le processus proposé ici, plus aucune conversion n'est nécessaire après celle exécutée entre les étapes 1 et 2 qui est automatique et sans approximation. Les différentes zones sont cloisonnées : le *designer* est le seul à pouvoir modifier les zones de subdivision et l'ingénieur le seul à pouvoir agir sur les vecteurs nodaux. La zone de raccord est gérée automatiquement par le modeleur et le maillage des zones NURBS est commun aux deux parties : chacun peut le modifier. Ce processus devrait permettre de trouver plus rapidement un compromis car le *designer* et l'ingénieur peuvent travailler en parallèle sur le modèle (sous réserve bien sûr de ne pas modifier la même zone en même temps) et toutes les étapes de conversion n'ont plus lieu d'être.

Ce processus n'est pas pour autant une solution miracle : premièrement les deux parties doivent agir sur le maillage des zones NURBS ce qui entraîne forcément des conflits. Egalement, le maillage n'est plus d'un seul bloc : chaque zone (subdivision, NURBS, et raccord) a son propre maillage. Ensuite en interdisant à l'ingénieur d'accéder aux zones de subdivision, certaines optimisation ne sont plus possibles. Enfin, malgré un contrôle global des lignes principales du modèle grâce aux zones de subdivision qui sont réparties aux jonctions de celles-ci, le *designer* n'a pas un contrôle total de la forme à cause des zones NURBS.

#### **Etape 4 : optimisation topologique**

Le fait d'utiliser le logiciel MODITERE dans le processus de conception permet d'ajouter, et sans le moindre effort supplémentaire, une étape d'optimisation topologique. Une fois le modèle validé, il faut le construire, ce qui signifie remplir cette surface extérieure par de la matière. Il est tout naturel de procéder à une étape optimisation de la matière pour réduire le poids de la pièce et économiser un coût de production.

Il est même possible d'imaginer cette étape d'optimisation comme une partie de l'étape de validation, car l'utilisation de surfaces lacunaires permet un *design* d'un autre genre et influence les calculs de l'ingénieur. Avec une surface extérieure trouée, ou une enveloppe transparente, il est également possible d'utiliser les structures intérieures comme partie intégrante du *design* du modèle.



# Chapitre 6

## Conclusion

Le formalisme des CIFS possède un certain nombre d'outils qui peuvent être utilisés sur n'importe quel type d'objet géométrique à la seule condition que celui-ci puisse être générer par un processus itératif. Jusqu'à présent, seules les fractales et les courbes et surfaces de Bézier et B-Splines uniformes étaient décrites dans le formalisme des CIFS. Au cours de cette thèse, de nouveaux objets géométriques ont été ajoutés à la collection des automates CIFS.

Dans un premier temps, les surfaces de subdivision ont été étudiées et de nombreux schémas de subdivision à support compact ont été représentés sous la forme d'automates CIFS. De plus, un protocole de génération de l'automate CIFS d'un schéma directement à partir des règles de subdivision a été déduit de ces différents exemples.

La seconde partie de cette thèse a consisté à représenter les NURBS, qui sont les courbes et surfaces les plus utilisées en CGAO, sous la forme d'automates CIFS. Deux méthodes ont été trouvées :

- la première utilise l'évolution du vecteur nodal au cours des étapes de subdivision pour en déduire la quasi-auto-similarité des NURBS et ainsi construire l'automate correspondant.
- la seconde consiste à utiliser le *blossoming* pour trouver le polygone de contrôle de la B-Spline uniforme qui correspond exactement à un morceau d'une NURBS.

Grâce à la représentation des surfaces de subdivision et des NURBS sous la forme d'automates CIFS, les outils du formalisme sont maintenant utilisables sur ces nouveaux objets géométriques. Parmi ces outils, celui permettant la création de raccords entre deux surfaces en conservant les propriétés de continuité de ces deux surfaces est particulièrement intéressant car il permet la création de surfaces mixtes dont la forme est gérée par plusieurs maillages de contrôle. Ainsi, il est possible de faire travailler un *designer* utilisant des surfaces de subdivision et un ingénieur utilisant des NURBS en parallèle sur le même modèle sans les conversions habituellement nécessaires entre chaque modification.

Pour finir, une architecture logicielle, appelée *shaders* de combinaisons barycentriques, permettant d'afficher en temps réel une surface décrite par un automate CIFS en passant par le *pipeline* OpenGL a été conçue. Celle-ci a ensuite été comparée, à la fois sur le plan de la mémoire utilisée et du temps de calcul nécessaire à l'affichage, à deux méthodes classiques de rendu de surfaces de subdivision. Il s'avère que la méthode proposée consomme un peu plus de mémoire que la méthode la moins lourde mais c'est la méthode la plus rapide pour le rendu de maillages animés.

## Travaux futurs

Plusieurs travaux peuvent être menés dans la continuité de cette thèse :

Premièrement, la gestion des automates CIFS correspondant à des schémas de subdivision à support non-compact. En analysant l'augmentation du nombre de positions possibles des irrégularités en fonction du degré, il est sûrement possible de trouver une méthode automatique de génération des différents états des automates CIFS correspondant aux schémas B-Splines uniformes de degrés élevés.

Deuxièmement la création des automates CIFS de schémas de subdivision non-uniformes (NURSS) [SZSS98, MRF06, CADS09]. Ceux-ci permettraient un compromis entre les surfaces de subdivision et les surfaces NURBS tensorielles. Des pistes de recherches ont été étudiées mais pour l'instant aucun résultat n'est suffisamment satisfaisant pour la publication.

Enfin, pour un réel impact sur l'industrie, il faudrait faire accepter le formalisme des CIFS par la communauté. Pour l'instant, les NURBS sont le modèle le plus utilisé et comme les outils de CGAO ont été optimisés pour ce modèle, l'industrie a une tendance à l'inertie. Pour faire accepter le modèle, il faut prouver que la transition des NURBS vers les modèles CIFS est rentable, c'est à dire que les gains apportés par le formalisme CIFS apporte plus que le coût de la transition industrielle.

# Bibliographie

- [ABC<sup>+</sup>13] Michele Antonelli, Carolina Vittoria Beccari, Giulio Casciola, Roberto Ciarloni, and Serena Morigi. Subdivision surfaces integrated in a CAD system. *Computer-Aided Design*, 45(11) :1294–1305, 2013.
- [AS02] Ergun Akleman and Vinod Srinivasan. Honeycomb subdivision. In *Proc. 17th Int. Symp. Computer & Information Sc*, pages 137–141, 2002.
- [Ban22] Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1) :133–181, 1922.
- [Bar14] Michael Barnsley. *Fractals everywhere*. Academic press, 2014.
- [Ben09] Hicham Bensoudane. *Etude différentielle des formes fractales*. PhD thesis, Université de Bourgogne, 2009. Thèse de doctorat dirigée par Neveu, Marc Informatique Dijon 2009.
- [Bli77] James F Blinn. Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH computer graphics*, volume 11, pages 192–198. ACM, 1977.
- [C<sup>+</sup>84] Georg Cantor et al. De la puissance des ensembles parfaits de points : Extrait d'une lettre adressée à l'éditeur. *Acta Mathematica*, 4 :381–392, 1884.
- [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. *ACM Transactions on Graphics (TOG)*, 28(3) :46, 2009.
- [CC78] Edwin Catmull and James Clark. Recursively generated B-Spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6) :350–355, 1978.
- [Cha74] George Merrill Chaikin. An algorithm for high-speed curve generation. *Computer graphics and image processing*, 3(4) :346–349, 1974.
- [CHR13] Thomas J. Cashman, Kai Hormann, and Ulrich Reif. Generalized Lane–Riesenfeld algorithms. *Computer Aided Geometric Design*, 30(4) :398 – 409, 2013.
- [Cox72] Maurice G Cox. The numerical evaluation of B-Splines. *IMA Journal of Applied Mathematics*, 10(2) :134–149, 1972.
- [CS66] H. B. Curry and I. J. Schoenberg. On pôlya frequency functions IV : The fundamental spline functions and their limits. *Journal d'Analyse Mathématique*, 17(1) :71–107, Dec 1966.
- [DK16] Neil A Dodgson and Jiří Kosinka. Can local NURBS refinement be achieved by modifying only the user interface ? *Computer-Aided Design*, 71 :28–38, 2016.
- [DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer Graphics and interactive techniques*, pages 85–94. Citeseer, 1998.
- [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.
- [DR47] Georges De Rham. Un peu de mathématiques à propos d'une courbe plane. *Elemente der Mathematik*, 2 :73–76, 1947.
- [DS78] Daniel Doo and Malcolm Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6) :356–360, 1978.
- [FJ98] Jean-Charles Fiorot and Pierre Jeannin. Une nouvelle construction du polygone massique de contrôle d'une courbe spline rationnelle. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 326(8) :1015 – 1019, 1998.

- [Gol04] Ron Goldman. The fractal nature of Bézier curves. In *Geometric Modeling and Processing*, 2004. Proceedings, pages 3–11. IEEE, 2004.
- [Hal01] Thomas C Hales. The honeycomb conjecture. *Discrete & Computational Geometry*, 25(1) :1–22, 2001.
- [Hau18] Felix Hausdorff. Dimension und äußeres maß. *Mathematische Annalen*, 79(1-2) :157–179, 1918.
- [Hil91] David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. *Mathematische Annalen*, 38 :459–460, 1891.
- [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*, pages 35–44. ACM, 1993.
- [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.
- [Kob00] Leif Kobbelt.  $\sqrt{3}$ -subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112. ACM Press/Addison-Wesley Publishing Co., 2000.
- [Koc04] HV Koch. Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire. *Arkiv for Matematik, Astronomi och Fysik*, 1 :681–704, 1904.
- [Loo87] Charles Loop. Smooth subdivision surfaces based on triangles. *Master’s thesis, University of Utah, Department of Mathematics*, 1987.
- [LR80] Jeffrey M Lane and Richard F Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1 :35–46, 1980.
- [Ma05] Weiyin Ma. Subdivision surfaces for CAD—an overview. *Computer-Aided Design*, 37(7) :693–709, 2005.
- [Man67] Benoit Mandelbrot. How long is the coast of britain ? Statistical self-similarity and fractional dimension. *science*, 156(3775) :636–638, 1967.
- [Man75] Benoit Mandelbrot. *Les objets fractals : forme, hasard et dimension*. Flammarion, 1975.
- [Men26] K Menger. Allgemeine räume und cartesische räume. i, communications to the amsterdam academy of sciences, 1926. english translation reprinted in edgar, gerald a., ed. 2004, classics on fractals. *Studies in Nonlinearity*, Westview Press. Advanced Book Program, Boulder, CO, 1926.
- [MGL<sup>+</sup>19] Lucas Morlet, Christian Gentil, Sandrine Lanquetin, Marc Neveu, and Jean-Luc Baril. Representation of NURBS surfaces by Controlled Iterated Functions System automata. *Computers & Graphics : X*, 2 :100006, 2019.
- [MGLN19] Lucas Morlet, Christian Gentil, Sandrine Lanquetin, and Marc Neveu. Uniformisation de NURBS par blossoming. *Journées Françaises d’Informatique Graphique 2019*, Marseille, 2019.
- [MNLG17a] Lucas Morlet, Marc Neveu, Sandrine Lanquetin, and Christian Gentil. Discréétisation directe de la surface limite de Catmull-Clark par Systèmes de Fonctions Itérés. In *Journées du Groupe de Travail en Modélisation Géométrique*, ENS Cachan, France, mars 2017.
- [MNLG17b] Lucas Morlet, Marc Neveu, Sandrine Lanquetin, and Christian Gentil. Calcul direct d’une tessellation de la surface limite pour les schémas de subdivision uniformes. In *Journées Françaises d’Informatique Graphique*, INRIA Rennes, France, octobre 2017.
- [MNLG18a] Lucas Morlet, Marc Neveu, Sandrine Lanquetin, and Christian Gentil. Représentation des NURBS par Systèmes Itérés de Fonctions. In *Journées du Groupe de Travail en Modélisation Géométrique*, ENSAM Aix-en-Provence, France, mars 2018.
- [MNLG18b] Lucas Morlet, Marc Neveu, Sandrine Lanquetin, and Christian Gentil. Barycentric Combinations Based Subdivision Shaders. In *Journal of WSCG*, volume 25, Plzen, Czech Republic, mai 2018.
- [MNLG18c] Lucas Morlet, Marc Neveu, Sandrine Lanquetin, and Christian Gentil. NURBS and Iterated Functions Systems. In *Curves & Surfaces*, Arcachon, France, juin 2018. Curves & Surfaces.
- [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.

- [Pea90] Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1) :157–160, 1890.
- [PGSL13] Sergey Podkorytov, Christian Gentil, Dmitry Sokolov, and Sandrine Lanquetin. Geometry control of the junction between two fractal curves. *Comput. Aided Des.*, 45(2) :424–431, February 2013.
- [Pod13] Sergey Podkorytov. *Espaces tangents pour les formes auto-similaires*. PhD thesis, Université de Bourgogne, 2013. Thèse de doctorat dirigée par Gentil, Christian et Sokolov, Dmitry Informatique Dijon 2013.
- [PR97] Jörg Peters and Ulrich Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics (TOG)*, 16(4) :420–431, 1997.
- [Ram87] Lyle Ramshaw. *Blossoming : A connect-the-dots approach to splines*. Digital Equipment Corporation Palo Alto, 1987.
- [Rei95] Ulrich Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2) :153 – 174, 1995.
- [RS00] Ulrich Reif and Peter Schröder. Curvature smoothness of subdivision surfaces. *Technical Report TR-00-03*, 2000.
- [Sch46] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions : Part A - On the problem of smoothing or graduation. A first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4(1) :45–99, 1946.
- [SKSD14] Jingjing Shen, Jiří Kosinka, Malcolm A Sabin, and Neil A Dodgson. Conversion of trimmed NURBS surfaces to Catmull–Clark subdivision surfaces. *Computer Aided Geometric Design*, 31(7-8) :486–498, 2014.
- [SKSD16] Jingjing Shen, Jiří Kosinka, Malcolm Sabin, and Neil Dodgson. Converting a CAD model into a non-uniform subdivision surface. *Comput. Aided Geom. Des.*, 48(C) :17–35, November 2016.
- [SL03] Jos Stam and Charles Loop. Quad/triangle subdivision. In *Computer Graphics Forum*, volume 22, pages 79–85. Wiley Online Library, 2003.
- [SLG05] Scott Schaefer, David Levin, and Ron Goldman. Subdivision schemes and attractors. In *Symposium on Geometry Processing*, pages 171–180. Citeseer, 2005.
- [Sta98a] 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*, pages 395–404. ACM, 1998.
- [Sta98b] Jos Stam. Evaluation of Loop subdivision surfaces. In *SIGGRAPH'98 CDROM Proceedings*. Citeseer, 1998.
- [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.
- [War95] Joe Warren. Subdivision methods for geometric design. *Unpublished manuscript*, November, 1995.
- [Wil78] Lance Williams. Casting curved shadows on curved surfaces. In *ACM Siggraph Computer Graphics*, volume 12, pages 270–274. ACM, 1978.
- [Zor00] Denis Zorin. Subdivision zoo. *Subdivision for modeling and animation*, Schröder, Peter and Zorin, Denis, pages 65–104, 2000.
- [ZS01] Denis Zorin and Peter Schröder. A unified framework for primal/dual quadrilateral subdivision schemes. *Computer Aided Geometric Design*, 18(5) :429–454, 2001.
- [ZT96] Chems Eddine Zair and Eric Tosan. Fractal modeling using free form techniques. In *Computer Graphics Forum*, volume 15, pages 269–278. Wiley Online Library, 1996.



## Annexe A

# Démonstration par Jean-Luc Baril

Nous posons  $n = 2d - 1$  pour  $d \geq 1$ . Soit  $\mathcal{M}_n$  l'ensemble des mots de longueur  $n \geq 1$  sur l'alphabet  $\{1, 2, \dots, n\}$ , i.e., tous les mots  $w = w_1 w_2 \dots w_n$  avec  $w_i \in \{1, 2, \dots, n\}$  for  $1 \leq i \leq n$ .  $w_i$  correspond à la  $i$ -ème lettre du mot  $w$  et  $\forall i, w_i \leq w_{i+1}$ . Le conjugué d'un mot  $w = w_1 \dots w_n$  est le mot  $\bar{w} = n + 1 - w_n, n + 1 - w_{n-1}, \dots, n + 1 - w_1$ .

A partir d'un mot  $w \in \mathcal{M}_n$ , nous construisons le mot  $W \in \mathcal{M}_{2n}$  défini par  $W = w_1 w_1 w_2 w_2 \dots w_n w_n$ . Notez que  $W_n = W_{n+1} = w_d$ . Aussi, nous définissons les transformations  $\mathcal{L}$  et  $\mathcal{R}$  sur  $\mathcal{M}_n$  de la manière suivante :

$$\begin{cases} \mathcal{L}(w_1 \dots w_n) &= W_{n-d+1} \dots W_{n-2} W_{n-1} W_n W_{n+1} \dots W_{n+d-1}, \\ \mathcal{R}(w_1 \dots w_n) &= W_{n-d+2} \dots W_{n-1} W_n W_{n+1} W_{n+2} \dots W_{n+d}. \end{cases}$$

Par exemple, quand  $n = 7$  (ou de manière équivalente  $d = 4$ ) et  $w = 1234567$ , nous avons  $W = 11223344556677$ ,  $\mathcal{L}(w) = 2334455$ ,  $\mathcal{R}(w) = 3344556$ ,  $\mathcal{L}(\mathcal{L}(w)) = 3334444$  et  $\mathcal{R}(\mathcal{L}(w)) = 3344445$ .

Pour  $n \geq 1$ , nous considérons l'ensemble  $\mathcal{S}_n$  des mots obtenus à partir de  $12 \dots n$  après toutes les combinaisons possibles de  $\mathcal{L}$  et  $\mathcal{R}$ . Par exemple,  $\mathcal{S}_7 = \{1234567, 2334455, 3344556, 3334444, 3344445, 3444455, 4444555, 3444444, 3344444, 4444445, 4444455, 4444444\}$ .

Les propriétés suivantes sont démontrées directement :

**Propriété 1.** Chaque mot  $w$  dans  $\mathcal{S}_n$  satisfait  $w_d = d$ .

**Propriété 2.** Pour  $1 \leq i \leq n - 1$ , deux lettres consécutives  $w_i$  et  $w_{i+1}$  dans le mot  $w \in \mathcal{S}_n$  on a toujours  $0 \leq w_{i+1} - w_i \leq 1$ .

**Propriété 3.** Pour chaque  $w \in \mathcal{S}_n$ , son conjugué  $\bar{w}$  repose également sur  $\mathcal{S}_n$ .

**Propriété 4.** (i)  $d^n$  est le seul mot  $w$  où  $\mathcal{L}(w) = \mathcal{R}(w) = w$ .

(ii)  $(d-1)^{d-1}d^d$  est le seul mot  $w$  où  $\mathcal{L}(w) = w$  and  $\mathcal{R}(w) \neq w$ .

(iii)  $d^d(d+1)^{d-1}$  est le seul mot  $w$  où  $\mathcal{R}(w) = w$  and  $\mathcal{L}(w) \neq w$ .

**Propriété 5.** Pour chaque  $w \in \mathcal{S}_n$ , le nombre de lettres  $d$  dans  $\mathcal{L}(w)$  (respectivement  $\mathcal{R}(w)$ ) est au minimum celui de  $w$ .

**Lemme 1.** Chaque mot  $w \in \mathcal{S}_n$  est un sous-mot  $m_k = 1^{2^k} 2^{2^k} \dots (n-1)^{2^k} n^{2^k}$  pour un  $k \geq 0$ .

Démonstration : Soit  $w$  un mot obtenu après l'application de  $r \geq 0$  transformations consécutives  $\mathcal{L}$  et/ou  $\mathcal{R}$ , et prouvons par induction sur  $r$  que  $w$  est un sous-mot de  $m_k = 1^{2^k} 2^{2^k} \dots (n-1)^{2^k} n^{2^k}$  pour un  $k$ . La preuve est évidente pour  $r = 1$  :  $w$  est un des deux mots  $\mathcal{L}(12 \dots n)$  et  $\mathcal{R}(12 \dots n)$ , qui sont des sous-mots de  $m_1 = 1122 \dots nn$  (par la définition même de  $\mathcal{L}$  et  $\mathcal{R}$ ). Supposons que  $w$  est un sous-mot de  $m_k$  avec  $w_d = d$ , et prouvons que  $\mathcal{L}(w)$  et  $\mathcal{R}(w)$  sont des sous-mots de  $m_{k+1}$ .

Par définition,  $w' = \mathcal{L}(w)$  (respectivement  $w' = \mathcal{R}(w)$ ) est obtenu comme un sous-mot de  $W = w_1 w_1 w_2 w_2 \dots w_n w_n$  après avoir aligné  $w'_d$  sur  $W_n = w_d$  (respectivement  $W_{n+1} = w_d$ ). Comme  $w$  est un sous-mot de  $m_k$  tel que  $w_d = d$ , cela implique que  $W = w_1 w_1 \dots w_n w_n$  est un sous-mot de  $m_{k+1}$ , et par conséquent  $w' = \mathcal{L}(w)$  (respectivement  $w' = \mathcal{R}(w)$ ) est un sous-mot de  $m_{k+1}$ .  $\square$

**Lemme 2.** Soit  $w$  un sous-mot de longueur  $n$  de  $m_k$ ,  $k \geq 0$ , tel que  $w_d = d$ . Alors,  $w$  appartient à  $\mathcal{S}_n$ .

Démonstration : La preuve est faite par induction de  $k$ . De manière évidente, quand  $k = 0$  l'unique sous-mot de longueur  $n$  de  $12\dots(n-1)n$  est  $12\dots n$ , et il appartient à  $\mathcal{S}_n$ . Pour  $k = 1$ ,  $w$  est un sous-mot de  $W = 1122\dots dd\dots nn$  avec  $w_d = W_n = W_{n+1} = d$ . Dans le cas où  $w_d$  correspond à  $W_n$ ,  $w = \mathcal{L}(12\dots n)$ ; sinon  $w_d$  correspond à  $W_{n+1}$  et nous avons  $w = \mathcal{R}(12\dots n)$ . Dans les deux cas,  $w$  repose sur  $\mathcal{S}_n$ .

Maintenant, supposons que le lemme est vrai pour  $i \leq k$ , et montrons qu'il est également vrai pour  $k + 1$ . Soit  $w$  un sous-mot de longueur  $n$  de  $m_{k+1}$  tel que  $w_d = d$ . Soit  $r \geq 1$  et donc  $w_d$  correspond à la  $r$ -ième valeur de  $m_{k+1}$ . Comme  $w_d = d$  nous avons forcément  $r \geq (d-1)2^{k+1} \geq n$ , et  $r \leq n2^{k+1} - (d-1)2^{k+1} < n2^{k+1} - n$ . Ces deux dernières inégalités nous assure qu'il existe un sous-mot  $W$  de longueur  $2n$  de  $m_{k+1}$  tel que  $W_n$  ou  $W_{n+1}$  correspond à  $w_d$ . Dans le cas où  $w_d$  correspond à  $W_n$ , alors nous définissons le sous-mot  $v$  de longueur  $n$  en supprimant toutes les valeurs de rang impair de  $W$ , et  $v$  satisfait  $\mathcal{L}(v) = w$ . Dans le cas où  $w_d$  correspond à  $W_{n+1}$ , alors nous définissons le sous-mot  $v$  de longueur  $n$  en supprimant toutes les valeurs de rang pair de  $W$ , et  $v$  satisfait  $\mathcal{R}(v) = w$ . Par construction,  $v$  est un sous-mot de  $m_k$  avec  $v_d = d$  et après avoir appliqué l'hypothèse de récurrence,  $v$  appartient à  $\mathcal{S}_n$ . Comme  $w$  est soit  $\mathcal{R}(v)$  soit  $\mathcal{L}(v)$ , nous déduisons que  $w$  est également dans  $\mathcal{S}_n$ . Nous concluons par induction.  $\square$

**Théorème 1.** Pour  $d \geq 1$ , le cardinal de l'ensemble  $\mathcal{S}_{2d-1}$  est  $4(d-1)$ .

Proof : Soit  $r \geq 1$  un entier tel que  $2^{r-1} < n = 2d - 1 < 2^r$ , i.e.  $r = \lceil \log_2(n) \rceil$ . En utilisant les deux lemmes précédents, il suffit de compter le nombre de sous-mots  $w$  de longueur  $n$  de  $m_k$ ,  $k \geq 0$ , satisfaisant  $w_d = d$ . Pour  $k = 0$  à  $r-1$ , le nombre de ces sous-mots dans  $m_k$  est  $2^k$  tant que le mot contient exactement  $2^k$  lettres  $d$ , et nous avons  $2^k$  manière de poser  $w_d = d$ .

Maintenant, comptons les sous-mots ayant exactement  $\alpha$  lettres  $d$  pour  $\alpha > 2^{r-1}$ . Il apparaît immédiatement que ce genre de mot est un sous-mot de  $m_r$  et pour  $\alpha < n$ , il y a deux manières de choisir ce mot (soit  $w_d = d$  est le  $d$  le plus à gauche de  $m_r$  soit  $w_d = d$  est le  $d$  le plus à droite de  $m_r$ ). Il y a donc  $2(n-1-2^{r-1})$  mots possibles quelque soit  $2^{r-1} < \alpha < n$ . Finalement, pour  $\alpha = n$ , il n'existe qu'un seul sous-mot qui est  $d^n$ .

En combinant tous ces cas, le nombre de mots de  $\mathcal{S}_n$  est :

$$\sum_{k=0}^{r-1} 2^k + 2(n-1-2^{r-1}) + 1 = 2^r - 1 + 2(n-1) - 2^r + 1 \quad (\text{A.1})$$

$$= 2(2d-2) = 4(d-1) \quad (\text{A.2})$$

$\square$

**Corollaire 1.** Pour  $12\dots n$ , nous pouvons atteindre n'importe quel mot de  $\mathcal{S}_n$  en  $\lceil \log_2(n) \rceil$  transformations  $\mathcal{L}$  et/ou  $\mathcal{R}$  ou moins. Il faut  $\lceil \log_2(n) \rceil$  transformations pour atteindre le mot  $d^n$ .

Proof : Soit  $r \geq 0$  l'entier tel que  $2^{r-1} < n < 2^r$ . Nous avons  $r = \lceil \log_2(n) \rceil$ . En utilisant le fait que  $d^n$  est obtenu comme un sous-mot de  $2^r$ , nous avons besoin de  $r$  transformations pour l'atteindre. A partir de la preuve du Théorème 1, tous les autres mots peuvent être atteints en  $r$  transformations ou moins, ce qui valide le corollaire.  $\square$

**Corollaire 2.** Soit  $w$  un sous-mot de  $\mathcal{S}_n$  tel que  $w \notin \{d^n, (d-1)^{d-1}d^d, d^d(d+1)^{d-1}\}$ . Il est impossible de trouver une séquence  $\xi$  de  $\mathcal{L}$  et  $\mathcal{R}$  telle que  $w = \xi(w)$ .

Proof : Comme  $w$  n'appartient pas à  $\{d^n, (d-1)^{d-1}d^d, d^d(d+1)^{d-1}\}$ , la propriété 4 prouve que  $\mathcal{R}(w)$  et  $\mathcal{L}(w)$  diffère de  $w$ . Soit  $r$  le nombre de lettres  $d$  dans  $w$ . Alors,  $\mathcal{L}(w)$  (respectivement  $\mathcal{R}(w)$ ) a au moins une lettre  $d$  de plus que  $w$  et comme  $\mathcal{L}$  and  $\mathcal{R}$  ne peuvent réduire le nombre de  $d$  (voir propriété 5), la preuve est faite.  $\square$

## Annexe B

# Notations

Dans cette annexe sont consignées l'ensemble des notations utilisées dans le manuscrit et leur signification. Pour plus de simplicité, celles-ci sont rangées par catégories. Il est conseillé de détacher cette annexe et de la garder en sous-main au cours de la lecture.

<u>Général</u>	
$\mathcal{C}, \mathcal{S}$	Courbe/Surface limite
$\mathbb{R}^N$	Espace euclidien de dimension $N$
$P = (\alpha_1 ; \dots ; \alpha_N)^\top$	Point de $\mathbb{R}^N$
$\mathbf{BI}^N$	Espace barycentrique de dimension $N$
$\mathbf{BI}^N(\omega)$	Espace barycentrique généralisé de dimension $N$ et de pondération $\omega$
$P = (\alpha_1 ; \dots ; \alpha_N ; \omega)$	Point de $\mathbf{BI}^N(\omega)$
<u>(C)IFS</u>	
$d$	Dimension fractale d'une fractale ou d'un (C)IFS
$n$	Nombre de transformations d'un (C)IFS
$\mathcal{T}_i$	Transformation numéro $i$
$\mathcal{M}_i$	Matrice associée à $\mathcal{T}_i$
$\mathcal{P}_i = \mathcal{T}_i^\infty$	Point-fixe de la transformation $\mathcal{T}_i$
$\vec{T}_i$ et $\vec{T}'_i$	Tangentes-principales de la courbe/surface en $\mathcal{P}_i$
$\vec{N}_i$	Normale à la courbe/surface en $\mathcal{P}_i$
$\mathcal{B}_i$	Point fixe de l'espace barycentrique
$\vec{\tau}_i$ et $\vec{\tau}'_i$	Tangentes-principales en $\mathcal{B}_i$ dans l'espace barycentrique
<u>Subdivision</u>	
$\mathbf{P}_0$	Polygone/maillage de contrôle
$\mathbf{P}_i$	Polygone/maillage après $i$ itérations
$\mathbf{P}_\infty$	Courbe/surface limite
$\mathbf{S}$	Opérateur de subdivision
$\mathbf{D}$	Opérateur de duplication de sommets
$\mathbf{M}$	Opérateur de moyennage
$s$	Nombre de sommets/arêtes d'une face
$k$	Valence d'un sommet
<u>NURBS</u>	
$d$	Degré de la NURBS
$\mathbf{T} = [t_0 = 0 \leq \dots \leq t_{2d-1} = 1]$	Vecteur nodal
$\mathbf{U} = [u_0 = t_1 - t_0 \dots u_{2d-2} = t_{2d-1} - t_{2d-2}]$	Vecteur inter-nodal
$\mathbf{V}$	Autre vecteur internodal pour les surfaces

