Modélisation de fruits, de leur structure interne et de leurs défauts

Evans Bohl

UNIVERSITÉ DE LIMOGES
ÉCOLE DOCTORALE « Sciences et Ingénierie pour l'Information, Mathématiques »
FACULTÉ DES SCIENCES ET TECHNIQUES
Laboratoire XLIM(DMI-SIR)

Thèse N°

THÈSE
pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ DE LIMOGES
Discipline / Spécialité : Informatique et Applications
présentée et soutenue par
Evans Bohl
le 4 Novembre 2015

Modélisation de fruits, de leur structure interne et de leurs défauts

Thèse dirigée par Olivier Terraz
et Djamchid Ghazanfarpour

JURY

Rapporteurs :	
M. David CAZIER	Professeur à l'Université de Strasbourg
M. Jean-Pierre JESSEL	Professeur à l'Université Paul Sabatier
Examinateurs :	
M. Philippe MESEURE	Professeur à l'Université de Poitiers
M. Olivier TERRAZ	Professeur à l'Université de Limoges
M. Djamchid GHAZANFARPOUR	Professeur à l'Université de Limoges

À tous mes proches, qui m'ont soutenu jusqu'au bout.

Remerciements

En tout premier lieu, je tiens à exprimer ma reconnaissance envers mes encadrants, M. Olivier Terraz ainsi que M. Djamchid Ghazanfarpour, à la fois pour leurs conseils et pour le temps qu'ils m'ont consacré. Un grand merci aussi pour m'avoir accordé leur confiance et pour m'avoir donné la possibilité de faire cette thèse dans de très bonnes conditions.

Je souhaite également exprimer ma gratitude à l'égard de M. David Cazier et M. Jean-Pierre Jessel pour avoir accepté d'être rapporteurs de mes travaux de thèse, ainsi qu'à l'égard de M. Philippe Meseure qui a accepté de faire partie du jury. J'espère que les travaux présentés ainsi que la rédaction seront à même de leur rendre la lecture de ce mémoire agréable.

Je tiens aussi à remercier toutes les personnes avec qui j'ai pu échanger longuement sur mon sujet de thèse, notamment mes collègues (Nico, Richard, Alexandre, etc...) et tous les autres. Cela m'a permis de prendre du recul sur ce que je faisais et d'appréhender différemment tous les problèmes rencontrés au cours de mes travaux.

Bien sûr, je n'oublie pas mes parents, ma soeur et toute ma famille, qui se trouvent à 16 000 kilomètres de Limoges. Malgré l'éloignement, vous m'avez toujours encouragé dans la voie que j'ai choisie. Vous avez été d'un soutien sans faille et c'est grâce à vous que je suis arrivé jusqu'au bout.

Résumé

La représentation de phénomènes naturels fait partie des domaines les plus complexes et les plus actifs de la recherche en informatique graphique. Notre compréhension de la nature s'améliorant au fil des années, les chercheurs ne cessent de proposer des nouveaux modèles, toujours plus pertinents les uns que les autres, et permettant de reproduire les différents phénomènes naturels que nous pouvons observer autour de nous, dans la vie de tous les jours.

Nous nous sommes intéressés à la représentation du fruit et des différents éléments qui le caractérisent. Le fruit est un objet complexe et, en fonction de la précision requise, sa conception à l'aide de logiciels de modélisation 3D peut très vite devenir compliquée. Notre modèle permet de générer une grande variété de fruits de formes différentes ainsi que les différents éléments de leur structure interne et ce, grâce à l'utilisation d'une seule grammaire. Au sein d'une même espèce, les fruits générés seront tous différents au niveau de leur forme, tout en restant semblables.

La seconde partie de nos travaux porte quant à elle sur la représentation des imperfections géométriques qui sont propres aux fruits. Les fruits sont le résultat de l'enchaînement d'un très grand nombre de processus physiologiques complexes qui interagissent fortement entre eux. Lorsque le bon fonctionnement de l'un de ces processus est compromis, cette anomalie se matérialise sur le fruit par l'apparition de défauts au niveau de sa forme. Notre modèle propose une approche simple, basée sur l'utilisation de grammaires, qui permettent d'altérer soit la forme générale d'un fruit soit des parties de sa surface.

Mots clefs : Synthèse d'images, modélisation procédurale, grammaires formelles, 3Gcartes L-systèmes, fruits.

Abstract

One of the largest areas of research in computer graphics deals with natural phenomena representation. Over the years, as our understanding of nature grew, researchers started to propose new ways of simulating the various natural phenomena that we can observe in our everyday life.

In this thesis, we focused on the representation of fruits. The fruit is a complex object. Depending on the desired accuracy, modeling a 3D fruit using classic 3D modeling software can become very tedious. We propose a model for generating vast varieties of fruits as well as their internal structure, thanks to the use of a single formal grammar. Each fruit that will be generated using our method will have global features that characterize its species, but it will also have local variations that are specific to it.

The second part of ou work involves the representation of geometrical imperfections along the fruit. The fruit is the result of a series of physiological processes that strongly interact with each other. When one of these processes does not work the way it should, this dysfunction is materialized in the form of a shape defect. Our model introduces a simple approach, based on the use of grammars, which will allow us to apply variations on fruits in order to generate various categories of shape defects.

Key words : Computer graphics, procedural modeling, formal grammars, 3Gmap L-systems, fruits.

Table des matières

Table des figures

Introduction

Le réalisme. Voici certainement l'un des principaux objectifs de la recherche en informatique graphique. En effet, ce domaine en constante évolution ne cesse de proposer des nouvelles méthodes, toujours plus efficaces les unes que les autres, permettant de reproduire le plus fidèlement possible différents phénomènes de la vie de tous les jours. Les applications de l'informatique graphique sont multiples et variées. Ainsi, grâce à l'évolution des ordinateurs, qui a permis d'effectuer de grands progrès ces dernières décennies, cette discipline est devenue peu à peu indispensable à l'industrie du loisir (effets spéciaux, jeux vidéo), au design industriel (conception et visualisation de prototypes) et à la réalité virtuelle (visites virtuelles interactives).

Parmi la multitude de thématiques de recherche qui existent en informatique graphique, la représentation de phénomènes naturels fait partie de celles qui sont non seulement les plus vastes et les plus complexes, mais aussi les plus jeunes. En effet, à l'aube de l'informatique graphique, très peu d'attention a réellement été donnée à la nature car elle était simplement trop complexe à simuler pour les ordinateurs de l'époque. Ainsi, plutôt que d'essayer de comprendre et de reproduire toute la complexité des phénomènes naturels, la majorité des efforts ont été concentrés dans un premier temps sur la reproduction virtuelle d'objets manufacturés par l'homme. Ensuite, à mesure que notre compréhension de la nature ainsi que la puissance de calcul offerte par les ordinateurs augmentaient, les chercheurs ont commencé à trouver des directions menant à la simulation de l'immense complexité de la nature.

Aujourd'hui encore, il est très difficile de comprendre toutes les facettes des phénomènes naturels que nous pouvons observer autour de nous. C'est pour cette raison que lorsque nous cherchons à simuler ces deniers, les modèles que nous développons sont souvent des versions simplifiés de la nature qui se contentent, dans un premier temps, de générer des résultats qui paraissent réels. Ensuite, à mesure que notre compréhension de la nature augmente, ces modèles finissent par s'enrichir, devenant ainsi de plus en plus fidèle à la réalité.

Problématique

La représentation des fruits et de leur phénomène de croissance est une thématique qui a été encore relativement peu abordée en informatique graphique. En effet, de par le fait que cet organe caractéristique des plantes à fleurs est le théâtre d'un très grand nombre de processus physiologiques qui interagissent fortement entre eux, la complexité de son fonctionnement fait que sa représentation peut très vite s'avérer complexe.

Les différents travaux présentés dans ce mémoire ont pour vocation de proposer une approche simple et intuitive de la représentation du fruit, de sa structure interne et des différents défauts pouvant affecter son aspect (malformations, maladies structurelles, etc. . .). Ainsi, le premier aspect abordé par ces travaux est la modélisation de la forme standard des différentes espèces de fruits. Pour ce faire, la méthode proposée doit pouvoir modéliser chaque détail de la forme d'un fruit selon la volonté de l'utilisateur, tout en respectant les différentes règles botaniques qui régissent sa forme.

Le second aspect abordé par nos travaux, directement relatif à la modélisation de la forme du fruit, est la génération de variété. En effet, les fruits d'une même espèce doivent tous présenter des caractéristiques qui sont propres à leur espèce, tout en présentant des variations qui leur sont propres et qui les rendent malgré tout différents les uns des autres. Ainsi, le modèle que nous allons introduire dans ce mémoire doit pouvoir donner la possibilité à l'utilisateur de générer plusieurs fruits d'une même espèce qui présentent donc des similarités sans pour autant être totalement identiques.

Enfin, la représentation d'imperfections est un aspect important de la représentation du fruit, car elle est directement liée à l'une des applications visées par nos travaux. En effet, bien que l'ajout d'imperfections sur des modèles 3D de fruits nous permettrait d'accroître le réalisme de ces derniers, un des autres avantages serait la possibilité de combiner notre modèle à des techniques de vision par ordinateur dans le but d'entraîner un système expert pour la reconnaissance de fruits. Pour ce faire, notre modèle doit être capable de créer une base de données contenant une grande variété de fruits 3D de formes différentes et d'aspects plus ou moins frais en fonction des défauts qu'ils présentent. Cette base de données pourra alors être injectée dans un système expert qui sera ainsi entraîné à trier des vrais fruits à l'issue de leur récolte, dans le but de pouvoir déterminer leur valeur commerciale.

Organisation du mémoire

Au cours du premier chapitre de ce mémoire, nous étudions les différents travaux en relation avec nos recherches. Dans la première partie de ce chapitre, nous présentons les grandes lignes de la biologie du fruit et de son développement. Nous présentons ensuite les différentes méthodes existantes en informatique graphique qui sont directement liées à la thématique du fruit. Puis, le concept de modélisation procédurale sera présenté, et nous finirons enfin par décrire les différents modèles utilisés tout au long de nos travaux, à savoir les 3G-cartes, les L-systèmes et les 3G-cartes L-systèmes.

Le second chapitre présentera notre modèle de génération de fruits et de leur structure interne. Nous introduirons ainsi dans un premier temps notre modèle de 3G-cartes L-système utilisé pour générer des fruits de formes variées et appartenant à des espèces différentes, grâce à l'utilisation d'une grammaire unique. La deuxième partie de ce chapitre montrera des extensions que nous avons introduit dans le modèle et qui permettent de générer la structure interne de nos modèles 3D de fruits.

Dans le dernier chapitre de ce mémoire, nous présenterons des outils permettant de générer des défauts géométriques sur des fruits, en nous focalisant sur le cas de la tomate. Dans la première partie de ce chapitre, nous présenterons une méthode permettant de faire varier la forme générale d'une tomate dans le but de générer des tomates mal formées. Ensuite, nous présenterons un ensemble d'extensions apportées au modèle, et qui donnent la possibilité à l'utilisateur de générer des variations géométriques locales. Nous allons ainsi présenter une méthode permettant de générer des excroissances. Puis nous montrerons comment générer des crevasses à la surface du fruit, et nous terminerons enfin par présenter une méthode pour créer des impacts de grêle et une méthode permettant de simuler l'affaissement d'une tomate.

Chapitre 1

État de l’art

Sommaire

Chapitre 1

État de l'art

La modélisation de phénomènes naturels fait partie des domaines de recherche les plus complexes et les plus actifs de l'informatique graphique. Il est possible d'en voir dans pratiquement tous les jeux vidéo ou films d'animation.

Dans cet état de l'art, nous nous intéressons à la représentation du fruit, de sa structure interne et de son vieillissement. Parmi le vaste nombre de domaines liées à la modélisation de phénomènes naturels, le fruit est un sujet qui a été encore relativement peu abordé en informatique graphique. Ainsi, afin de mieux appréhender les problématiques liées à ce sujet, nous avons jugé nécessaire de présenter dans la section 1.1 les grandes lignes de la biologie du fruit.

Les différents travaux s'intéressant à la représentation du fruit tentent de répondre à des problématiques aussi variées que complexes. Certaines méthodes font par exemple appel à la modélisation procédurale pour gérer l'organisation des organes au sein d'un fruit durant la croissance, alors que d'autres utilisent des modèles de réaction-diffusion pour simuler son vieillissement. Ainsi, dans la section 1.2 nous présenterons un survol des techniques de représentation de fruits en informatique graphique, tout en essayant de mettre en évidence leurs forces et leurs faiblesses.

Notre but étant de modéliser une grande variété de fruits de manière relativement simple, la modélisation procédurale s'est avérée être la méthode la plus adaptée à nos besoins. Nous présenterons donc dans la section 1.3 une étude des méthodes procédurales existantes, et terminerons ce chapitre par la section 1.4 dans laquelle nous présenterons les différents outils utilisés dans nos méthodes, à savoir, les 3G-cartes, les L-systèmes et les 3G-cartes L-systèmes.

1.1 Généralités sur les fruits

Le fruit est un objet complexe qui, selon Génard *et al.* [GBB+07], peut être vu comme un système au sein duquel une multitude de processus interagissent fortement entre eux (transpiration, photosynthèse, etc...). Il faudrait à ce titre plusieurs centaines de pages pour en appréhender tous les aspects. Ainsi, afin de ne pas sortir du cadre fixé par notre étude, nous n'aborderons ici que les aspects qui nous seront utiles à la compréhension de ce mémoire. Dans un premier temps, nous présenterons donc l'anatomie d'une fleur ainsi que le rôle qu'elle joue dans la formation du fruit. Ensuite, nous décrirons le cycle de vie du fruit ainsi que les principales catégories de fruits existantes. Puis, la structure interne d'un fruit sera présentée en nous focalisant sur ses aspects macroscopiques. Les aspects microscopiques ne seront pas abordés, car ils ne présentent que peu d'intérêts pour les travaux présentés ici. Enfin, nous terminerons par une présentation des défauts de fruits : caractéristiques qui affectent leur aspect, leur comestibilité ou leur valeur commerciale. Pour une présentation plus complète de la biologie du fruit, nous invitons le lecteur à se référer au livre de James D. Mauseth intitulé *Botany : An Introduction to Plant Biology* [Mau14], à partir duquel toutes les informations présentées ici ont été tirées.

1.1.1 Anatomie d'une plante à fleur

En botanique, le fruit est la caractéristique principale des Angiospermes, division qui regroupe les plantes à fleurs. Il résulte de la transformation de l'appareil reproducteur femelle après fécondation, et a pour rôle de favoriser la reproduction de l'espèce en protégeant la ou les graines qu'il contient. Avant de se transformer en graines (à l'issue de la fécondation), les ovules sont contenus dans une enveloppe protectrice d'origine foliacée : le carpelle. Cette pièce florale est composée de trois parties principales (Figure 1.1) :

- Le **stigmate**, qui est visqueux afin de pouvoir capter le pollen lors de la pollinisation ;
- Le **style**, en forme de colonne, permettant d'élever le stigmate à une position qui facilite la captation du pollen ;
- L'**ovaire**, situé à la base du style, qui abrite les ovules. L'ovaire est qualifié d'uniovulé lorsqu'il ne contient qu'un ovule et de pluriovulé dans le cas contraire.

Selon son espèce, une fleur peut contenir un ou plusieurs carpelles. Ces derniers constituent le **pistil**, organe femelle de la fleur. Un pistil est dit **gamnocarpellé**, lorsque ses carpelles sont soudés entre eux. Si ce n'est pas le cas, c'est un pistil **dialycarpellé**. Enfin, lorsque le pistil n'est composé que d'un unique carpelle, on dit qu'il est **unicarpellé**. Dans la nature, une grande majorité des angiospermes (un peu plus de 80%) ont des pistils gamnocarpellés. Nous allons voir plus loin que la variété d'un pistil a son importance, car elle permet de déterminer le type de fruit qui résultera de sa fécondation.

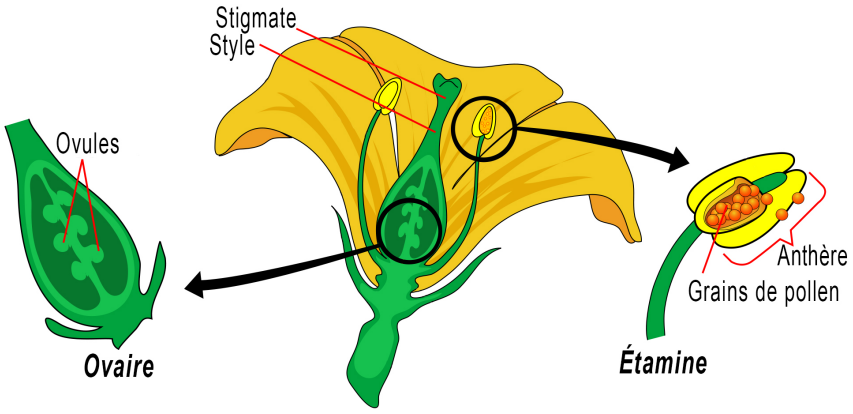


FIGURE 1.1 – Organes mâles et femelles d'un angiosperme (Illustration partagée par Mariana Ruiz Villarreal - commons.wikimedia.org (public domain))

1.1.2 Cycle de vie d'un fruit

Le cycle de vie d'un fruit peut être décomposé en quatre étapes. Dans un premier temps intervient la fécondation. Au cours de ce processus, les grains de pollen, cellules reproductrices mâles, sont transportés depuis les étamines (organes mâles de la fleur) vers le pistil. Lorsqu'un grain de pollen se dépose sur le stigmate, il germe et émet un tube pollinique qui lui permet de conduire les spermatozoïdes le long du style, afin de venir féconder les ovules qui sont contenus dans les ovaires.

Une fois la fécondation terminée, celle-ci est immédiatement suivie de la fanaison et de la chute de l'ensemble des pièces florales hormis les ovaires. C'est aussi au cours de cette étape qu'ont lieu un grand nombre de divisions cellulaires, dans le but de mettre en place la structure définitive du fruit. Lorsque cette étape est terminée, une expansion cellulaire permet d'augmenter sa taille. Dans certains cas, le fruit peut croître jusqu'à devenir cent fois plus grand que sa taille d'origine.

Enfin, une fois qu'il a atteint sa taille définitive, le fruit entre dans une phase de maturation. C'est au cours de cette étape finale que son rôle évolue. En effet, les embryons et les graines étant arrivés à maturité, le fruit n'a plus besoin de les protéger et doit donc les disperser de manière à pouvoir donner naissance à de nouvelles plantes. Il va alors changer de couleur, devenir plus juteux et sucré afin d'attirer les animaux qui vont le manger et répandre les graines dans la nature. Dans certains cas, le fruit ne sera pas mangé, et va donc pourrir afin de libérer les graines pour favoriser la reproduction. La figure 1.2 illustre les différentes étapes du cycle de vie de la tomate cerise.

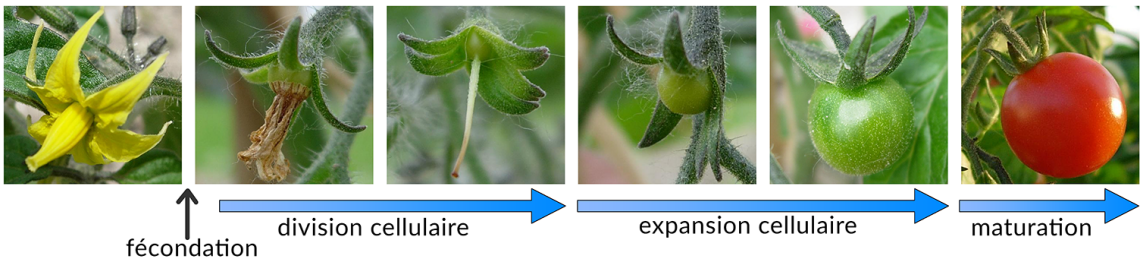


FIGURE 1.2 – Photographies illustrant les différentes étapes de la formation d'une tomate cerise, de la fleur au fruit (Photos partagées par Toony - commons.wikimedia.org (Creative Commons BY-SA 3.0))

1.1.3 Taxonomie des types de fruits

Dans les sections précédentes, nous avons vu que dans la plupart des cas, le fruit provient exclusivement de la transformation des parois de l'ovaire après fécondation. Il existe cependant quelques espèces chez qui d'autres parties de la fleur (comme le réceptacle) peuvent intervenir lors de la croissance. Ainsi, en fonction des organes impliqués et de la variété du pistil, il est possible de classer les fruits dans des catégories bien précises. Il en existe un très grand nombre, mais nous ne présenterons ici que ceux qui seront utiles à la compréhension de ce mémoire. Pour une présentation complète de toutes les catégories de fruits, nous invitons le lecteur à se référer à [Mau14].

Les fruits simples charnus
Ces fruits à la paroi épaisse et charnue sont issus d'un pistil qui peut être unicarpellé ou gamnocarpellé. Ils représentent environ 90% des fruits présents dans la nature, et il est possible d'en distinguer deux sous types selon la nature des couches de leur paroi. Lorsque toutes les couches sont charnues et que les graines (pépins) sont libres, il s'agit d'une baie (exemples : la tomate et le raisin). Lorsque les pépins sont contenus dans une couche interne ligneuse (de la nature du bois), le fruit est une drupe (exemples : la cerise, la pêche et l'abricot).

Les fruits multiples
Dans certains cas, les ovaires d'un pistil sont libres (pistil dialycarpellé) et finissent par fusionner lors de la croissance pour ne former qu'un seul et même fruit. On parle alors de fruit multiple. La framboise et la mûre en sont des exemples. Chez ces deux fruits chacune des drupéoles, petites boules gorgées d'eau et de sucre facilement identifiables, est en fait un fruit simple charnu.

Les fruits composés
Ces fruits ont la particularité d'être issus de plusieurs fleurs distinctes, toutes regroupées autour de la tige selon une architecture particulière, appelée inflorescence. Lors de la croissance, chaque fleur donnera un fruit qui viendra fusionner avec ses voisins pour ne former qu'une seul entité appelée infrutescence. L'ananas en est un exemple, et chacun de ses écussons est en fait un fruit simple provenant d'une seule fleur de l'inflorescence.

Les fruits complexes
Aussi appelés faux-fruits, ils sont issus de la transformation d'autres parties de la fleur en plus des ovaires. La pomme par exemple, est issue d'une fleur dont l'ovaire est complètement enfoncé dans le réceptacle florale. Ainsi après fécondation, ce dernier devient charnu et forme la partie comestible de la pomme, alors que le "vrai" fruit devient en fait le trognon qui n'est pas consommé. Chez la fraise également, la pulpe rouge et juteuse correspond en fait au réceptacle floral tandis que les petites graines croquantes sont les vrais fruits (appelés akènes).

1.1.4 Structure d'un fruit

La paroi d'un fruit, aussi appelée **péricarpe**, est issue de la transformation de la paroi des ovaires après fécondation. Le péricarpe a pour rôle de favoriser la reproduction de l'espèce en protégeant les graines issues de la transformation des ovules. Il est composé de trois couches distinctes :

- L'**exocarpe** : partie externe qui forme la peau colorée du fruit ;
- Le **mésocarpe** : partie intermédiaire qui représente la pulpe juteuse ;
- L'**endocarpe** : couche interne qui tapisse la loge où se trouvent les graines.

La figure 1.3 illustre la structure de deux fruits simples charnus : une baie et une drupe. À droite, nous pouvons voir la coupe transversale d'une pêche. Le pistil de la fleur de pêcher est uniovulé et unicarpellé, ce qui explique le fait que la pêche ne contient qu'une seule graine enveloppée dans un noyau. La partie gauche de la figure 1.3 illustre la coupe équatoriale d'une tomate ronde. Cette tomate est composée de cinq loges (donc cinq ovaires) contenant plusieurs graines. De plus les ovaires sont soudés entre eux et ne forment qu'une seule et même entité. Nous pouvons donc déduire que ce fruit est issu d'un pistil gamnocarpellé et pluriovulé.

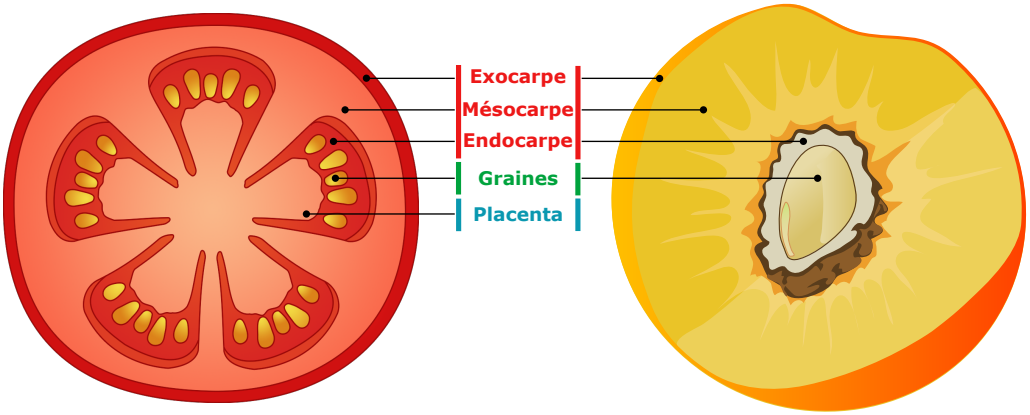


FIGURE 1.3 – Illustration de la structure d'une baie et d'une drupe : la tomate et la pêche.

Grâce aux informations présentées jusqu'à présent, nous pouvons affirmer que tous les fruits ont généralement une structure interne relativement similaire. En effet, hormis les fruits complexes, tous découlent exclusivement de la transformation des ovaires et des ovules à l'issu de la fécondation. De plus, 90% des fruits sont des fruits simples charnus, et les fruits multiples et composés consistent en fait en un agglomérat de fruits simples issus d'une seule fleur ou de plusieurs fleurs distinctes. Ainsi, en partant de ce constat, et à partir du moment où nous connaissons l'anatomie d'un fruit simple, il est assez aisé de se faire une représentation de la structure interne d'un fruit composé ou d'un fruit multiple.

C'est pour cette raison que nous n'avons présenté dans cette section que l'anatomie des fruits simples charnus. Celle des fruits complexes n'a pas non plus été traitée ici, car chaque espèce de ce type possède une structure qui lui est spécifique. De plus, à cause du fait qu'ils ne représentent qu'un faible nombre comparé aux autres catégories, les fruits complexes n'ont pas été traités au cours des travaux présentés dans ce mémoire. Néanmoins, leur modélisation constituera une extension future de nos travaux.

Dans le chapitre suivant, nous verrons que la relative similarité qui existe entre les structures internes de la majorité des fruits est importante. En effet, en fonction de cette caractéristique, nous avons identifié les 3G-cartes L-systèmes comme étant un outil intéressant permettant de pallier la majorité des contraintes du développement du fruit et de sa structure interne.

1.1.5 Défauts de fruits

Pour qu'un fruit soit apte à être commercialisé sur les étalages de supermarchés, celui-ci doit présenter un certain nombre de caractéristiques bien précises. Ces caractéristiques sont définies par des normes, elles-mêmes établies par deux organismes : la Commission économique pour l'Europe des Nations unies (CEE-ONU) et l'Organisation de coopération et de développement économiques (OCDE). Pour le contrôle de la qualité des tomates par exemple, ces normes sont définies dans [UNE12] et [OEC02]. Ces normes ont donc pour objet de définir les qualités que doit présenter un fruit lors de sa récolte afin de pouvoir être considéré comme étant propre à la consommation. Ainsi, en se basant sur ces standards, nous pouvons définir un défaut de fruit comme étant une caractéristique qui affecte l'aspect, la comestibilité ou la valeur commerciale de celui-ci.

Au cours de son cycle de vie, plusieurs facteurs peuvent altérer l'intégrité d'un fruit et générer ainsi des défauts de croissance. Chez la tomate par exemple, des crevasses peuvent apparaître à sa surface à cause d'une croissance trop rapide, d'un éclatement au cours d'une manipulation brutale ou même d'une attaque de grêle (figure 1.4a). Dans d'autres cas, le fruit peut avoir subi des dommages causés par des maladies ou des conditions climatiques défavorables (figure 1.4b). Chez l'orange par exemple, l'un des agents de pourriture que l'on retrouvera le plus souvent est le Penicillium digitatum (figure 1.4c), alors que la pomme quant à elle sera souvent infectée par le Neofabraea (figure 1.4d), tous les deux étant des espèces de champignons microscopiques.

(a) (b) (c) (d)

FIGURE 1.4 – (a) une crevasse fraîche sur une tomate ([OEC02]), (b) dégâts causés par le gel sur une tomate ([OEC02]), (c) une orange atteinte par le Penicillium digitatum (copyright Karlla Patricia, http ://diariode-biologia.com/), (d) une pomme atteinte de Neofabraea (copyright Patriik Mlčoch, houbyolbramic.blog.cz).

Ainsi, à cause du fait que chaque espèce de fruit présente ses propres défauts de croissance, nous pouvons voir qu'il existe une énorme quantité de défauts possibles, et il serait donc difficile de vouloir tous les représenter. C'est pour cette raison qu'au cours de nos travaux de recherche, bien que ces derniers nous ont permis de modéliser plusieurs espèces de fruits différentes, nous avons décidé de focaliser nos efforts sur la représentation d'une seule d'entre elles et des défauts affectant sa géométrie : la tomate. Cette espèce de

fruit présente l'avantage d'être commercialisée sous différentes variétés, chacune présentant ses propres caractéristiques, et nous verrons par la suite qu'à l'aide de notre modèle, il est possible de modéliser chacune de ces variétés de manière relativement simple.

1.2 Les fruits en informatique graphique

Jusqu'à présent, le fruit représente encore une thématique qui a été très peu abordée dans le domaine de l'informatique graphique, notamment à cause de sa complexité. En effet, l'énorme diversité des fruits et des maladies propres à chaque espèce, fait que la création d'outils capables de les représenter de manière générique et exhaustive est un véritable défi. De plus, dans le domaine de la botanique, beaucoup d'aspects du développement du fruit n'ont pas encore été totalement compris, et les personnes travaillant sur ce sujet tentent d'établir, aujourd'hui encore, des modèles mathématiques permettant de mieux comprendre la complexité de ce processus (exemple : le fruit virtuel de Génard *et al.* [GBB+07] illustré sur la figure 1.5).

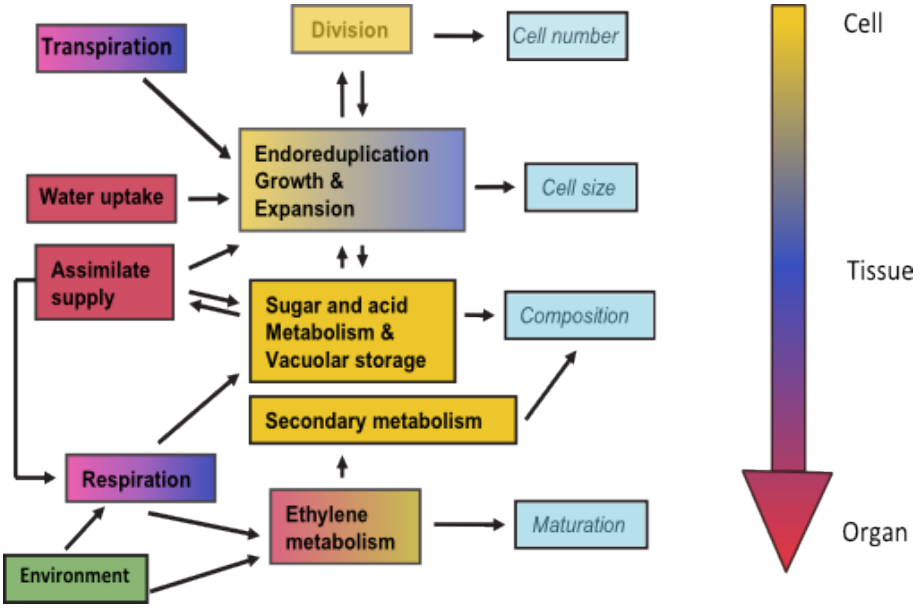


FIGURE 1.5 – Schéma du modèle botanique de Génard *et al.* [GBB+07], le fruit virtuel, qui tente de comprendre les processus qui interviennent au cours du développement du fruit.

Nous verrons dans la première partie de cette section les différentes méthodes qui ont permis de représenter des fruits "parfaits", c'est-à-dire sans signes de vieillissement apparents. La première méthode que nous présenterons ici sera présentée en détails, car elle est basée sur une approche relativement similaire à la nôtre. De plus, il s'agit des seuls travaux que nous avons identifiés comme étant axés spécifiquement sur la reproduction des phénomènes observés lors de la croissance de vrais fruits. Les autres méthodes quant à elles ne cherchent pas vraiment à reproduire des phénomènes biologiques, mais sont quand même intéressantes à introduire car elles présentent des fruits parmi leurs résultats. Ensuite, nous présenterons dans la seconde partie de cette section un état de l'art des méthodes permettant de simuler le vieillissement du fruit. Ici encore, nous décrirons plus

en détails la méthode qui nous a semblé la plus abouti dans le domaine, et qui propose une approche relativement similaire à la nôtre.

1.2.1 Représentation de fruits et de leurs structure interne

1.2.1.1 Modélisation procédurale

En botanique, l'inflorescence de la grappe de raisin est appelée panicule, car celle-ci est formée par une grappe de grappes sur un axe simple, appelé rachis (voir figure 1.6). La difficulté lorsqu'on veut modéliser une grappe de raisin réside dans le fait qu'il faut prendre en compte les interactions entre chacune de ses baies. En effet, à cause de son volume et de son poids, chaque baie va entrer en collision avec ses voisines lors de la croissance et va ainsi les repousser.

Pour répondre à cette problématique, Huang *et al.* [HJT+13] proposent d'utiliser une méthode procédurale basée sur les L-systèmes ouverts de Mech *et al.* [MP96]. Ces derniers sont une variante des L-systèmes [Lin68], un concept qui sera présenté plus en détails dans la suite de ce mémoire, et permettent de modéliser des objets naturels complexes à l'aide de règles définies dans une grammaire, tout en prenant en compte l'interaction avec l'environnement. Grâce à cette méthode, chaque baie est capable lors de sa croissance de connaître la position de ses voisines et d'optimiser ainsi la sienne.

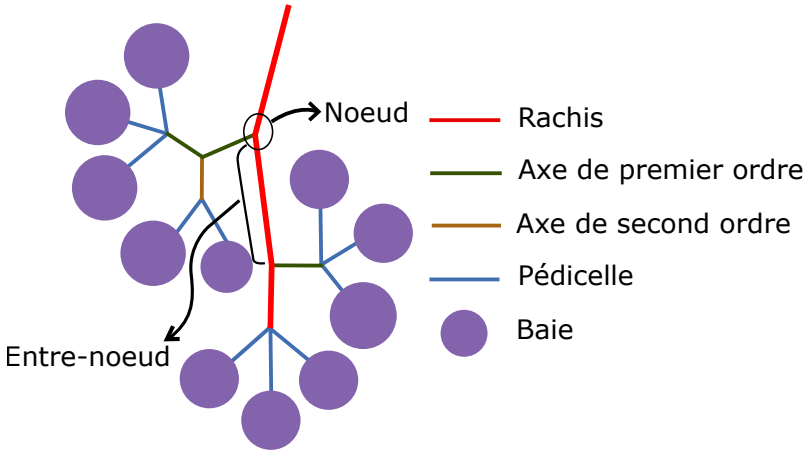


FIGURE 1.6 – Illustration des différents éléments qui composent une grappe de raisin.

Le processus de modélisation de [HJT+13] se déroule en trois étapes. Dans un premier temps l'utilisateur doit définir la forme de la grappe à modéliser à l'aide d'un polyèdre à huit faces (figure 1.7a). Il existe six types de formes de grappes (conique courte, cylindrique, conique épaulée, etc...) et selon les auteurs, les six points de contrôle d'un polyèdre à huit faces suffisent pour contrôler la forme et obtenir un résultat réaliste. Ainsi, la distance entre les points $p1$ et $p2$ du polyèdre détermine la longueur l du rachis, alors que la largeur w des axes de premier ordre est représentée par les points $p3$-$p6$ (figure 1.7b).

Lorsque le polyèdre a été mis en place, ses dimensions permettent de définir les paramètres du rachis et des axes de premier ordre. Le nombre de segment m du rachis est en général égal à la longueur du polyèdre (exemple : $m = 8$ pour le polyèdre de longueur 8 illustré sur la figure 1.7c). Le paramètre n quant à lui définie le nombre de segments du

plus long axe de premier ordre. En effet, les axes de premier ordre distribués le long du rachis n'ont pas tous la même longueur. L'axe le plus proche de la base du rachis (proche de $p1$) est celui qui contient le plus de segments et, plus on se rapproche de l'extrémité du rachis (point $p2$), plus le nombre de segments diminue par axe, avec un nombre minimum égal à 1 (figure 1.7c). Enfin, le paramètre dn permet de définir le taux de diminution du nombre de segments d'un axe à l'autre en fonction de sa position au sein du rachis.

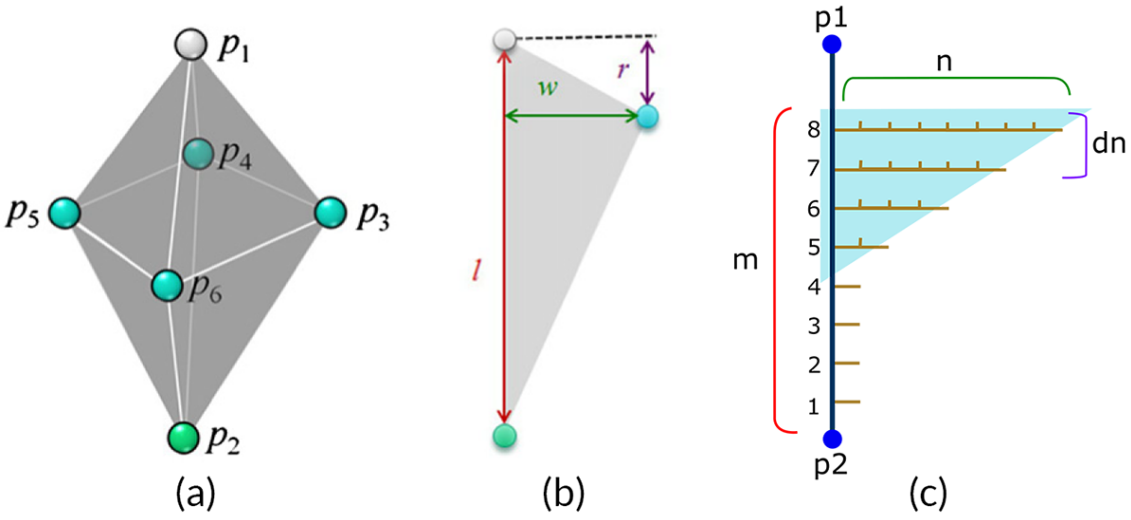


FIGURE 1.7 – Illustration des deux premières étapes de l'algorithme de [HJT+13]. (a) le polyèdre définissant la forme générale de la grappe, (b) le paramétrage du rachis et des axes de premier ordre et (c) un exemple de paramétrage.

Lorsque les paramètres du rachis et des axes de premier ordre ont été définis, le L-système fait croître la grappe en simulant à chaque étape les interactions qui ont lieu entre ses baies. Pour ce faire, deux caractéristiques sont prises en compte : l'interprétation par le bas et l'élagage. L'interprétation par le bas consiste à interpréter le L-système en partant des baies pour remonter vers la racine du rachis. Selon les auteurs, cette méthode permettrait de faciliter l'approximation de la gravité sur les baies. L'élagage quant à lui consiste en la réduction du nombre de baies dans la grappe. En effet, lorsqu'une grappe de raisin arrive à maturité, seulement une partie de ses bourgeons se sont transformés en baies. Cela est tout simplement dû au fait que ces dernières ne se développeront pas dans des endroits où elles n'auront pas assez d'espace. Ainsi, en respectant ces deux caractéristiques, les étapes de l'exécution du système sont donc :

1. Créer le noeud $A_r(0)$ de l'extrémité du rachis.
2. Générer un paquet de baies sur $A_r(0)$.
3. Allonger le rachis en y ajoutant un segment partant de $A_r(0)$ et donnant sur $A_r(1)$.
4. Créer un axe de premier ordre A_p partant de $A_r(1)$ et qui est composé de n segments, n ayant été calculé pour cet étage spécifique lors de la phase de paramétrage.
5. Créer un axe de second ordre A_s au niveau de chaque noeud de A_p.
6. Pour l'extrémité de chaque axe A_s et celle de l'axe A_p, tester l'espace pour voir si il est possible d'y créer un paquet de baies. Si oui, créer un paquet de baies. Sinon,

essayer d'ajuster la direction des axes pour pouvoir générer un paquet de baies, et si ce n'est toujours pas possible, ne rien faire.

7. Répéter les étapes 3 à 6 jusqu'à ce qu'on soit remonté à la racine du rachis. Lorsque c'est le cas, l'exécution du système est terminée.

La figure 1.8 illustre un exemple de résultat obtenu à l'aide de cette méthode, et pour des paramètres $m = 3$, $n_1 = 1$ et $n_2 = 2$.

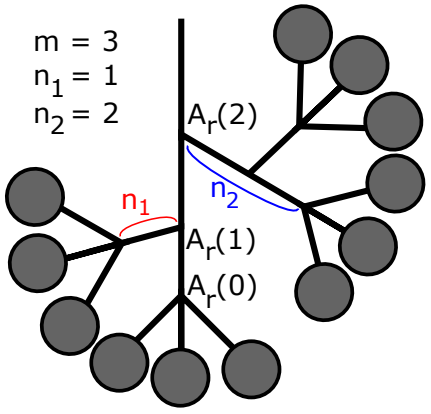


FIGURE 1.8 – Résultat obtenu par l'application de la méthode de [HJT+13]. Ici la taille du rachis est égale à trois unités de mesure.

Les résultats obtenus par la méthode de [HJT+13] sont très convaincants (figure 1.9). Ils permettent en effet d'obtenir des grappes de raisin de forme réaliste, et plus généralement des grappes d'autres espèces de baies (litchis, longanes, etc...). De plus, le choix d'une méthode procédurale couplée au contrôle de la forme par l'intermédiaire d'un polyèdre est très judicieux, et permet ainsi d'éviter un problème souvent rencontré chez les méthodes procédurales : le manque de contrôle sur le résultat final.

Il existe cependant un certain nombre de points qui pourraient être améliorés dans une éventuelle extension. Par exemple, dans le but de faciliter le contrôle des intersections entre les baies, les auteurs ont limité la forme de celles-ci à des sphères. Ainsi, une extension pourrait utiliser des baies de formes plus complexes, comme des tomates par exemple, et créer ainsi des grappes ayant l'air encore plus réalistes. Une autre extension possible serait l'utilisation de baies comportant une structure interne, le tout couplé à un modèle de vieillissement physiquement réaliste. Il serait alors possible de simuler par exemple le phénomène de flétrissement des baies que nous pouvons observer sur des vrais grappes de raisin.

FIGURE 1.9 – Des vraies grappes de raisin (gauche) comparées à des modèles obtenus par [HJT+13] (droite).

1.2.1.2 Représentation volumique

En informatique graphique, les maillages surfaciques sont le modèle de représentation d'objets 3D le plus répandu. Il sont très utilisés car ils sont compacts, ce qui facilite grandement leur construction et leur rendu. Cependant, ce modèle est limité pour certaines applications car il ne contient aucune information sur la structure interne de l'objet qu'il représente, rendant ainsi impossible la découpe de ce dernier pour inspecter son intérieur. Pour pallier ce défaut, les méthodes de représentation volumique ont été introduites. Contrairement aux méthodes de représentation surfacique, ces dernières permettent de créer des modèles ayant une structure interne que l'utilisateur peut explorer en les découpant en deux. Ici, les méthodes que nous présenterons ne sont pas, à proprement parlé, axées sur la thématique du fruit. Ainsi, elles se limitent uniquement à la représentation des fruits et de leur structure interne parmi leurs résultats, sans prendre en compte les différents processus biologiques qui interviennent lors de la croissance.

Les illustrations volumétriques ont été introduites par Owada *et al.* [ONOI04], puis améliorées par Pietroni *et al.* [POB+07]. Plutôt que d'utiliser des textures 3D, ce modèle de représentation permet d'assigner des textures 2D à l'intérieur d'un maillage surfacique, ce qui permet de créer des modèles qui sont compacts, tout en ayant des informations sur leur structure interne. Pour ce faire, l'utilisateur n'a qu'à définir des textures pour différentes coupes du maillage. Ainsi, à chaque fois que ce maillage sera coupé en deux, le système calculera une interpolation de toutes les textures 2D internes pour afficher la texture correspondante à la coupe. Bien que ce modèle de représentation soit compact, la qualité de la texture qui est synthétisée à chaque découpe peut manquer de précision, ce qui peut poser problème dans certains cas. De plus, étant donné le fait qu'on utilise toujours un maillage surfacique, la structure interne n'est pas vraiment modélisée, et il est par exemple impossible de faire du rendu translucide pour mettre en évidence sa structure interne. Parmi les résultats présentés, Owada *et al.* ont utilisé leur méthode pour découper des kiwis (figure 1.10), alors que Pietroni *et al.* représentent des oranges (figure 1.11).

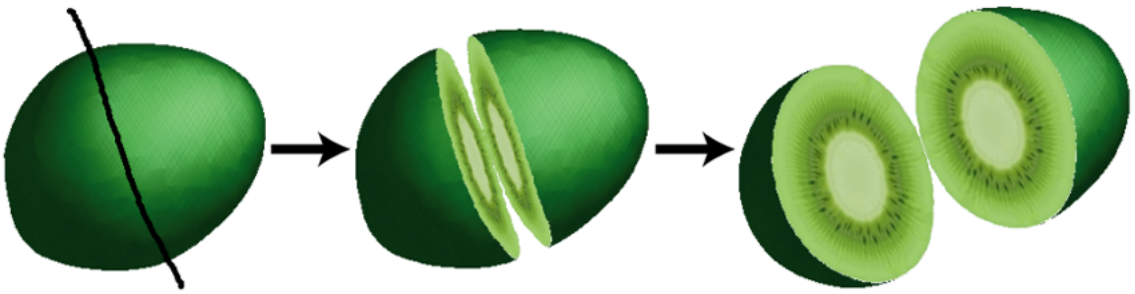


FIGURE 1.10 – Découpe d'un maillage de kiwi dont l'intérieur a été texturé par [ONOI04].

Takayama *et al.* [TOII08] introduisent les *lapped solid texutres* comme un analogue 3D aux *lapped textures* de Praun *et al.* [PFH00]. Ce concept consiste à appliquer des petites textures solides (ou textures 3D) à une tétraédrisation de l'intérieur d'une surface 3D. Chaque tétraèdre peut ainsi être vu comme l'équivalent d'un "patch" de [PFH00] en trois dimensions. Et lorsque l'utilisateur coupe l'objet en deux, la nouvelle surface ainsi créée intersecte la tétraédrisation qui a été préalablement texturée, donnant alors l'illusion d'une texture 3D continue. Cette approche présente deux avantages principaux. Premièrement, les

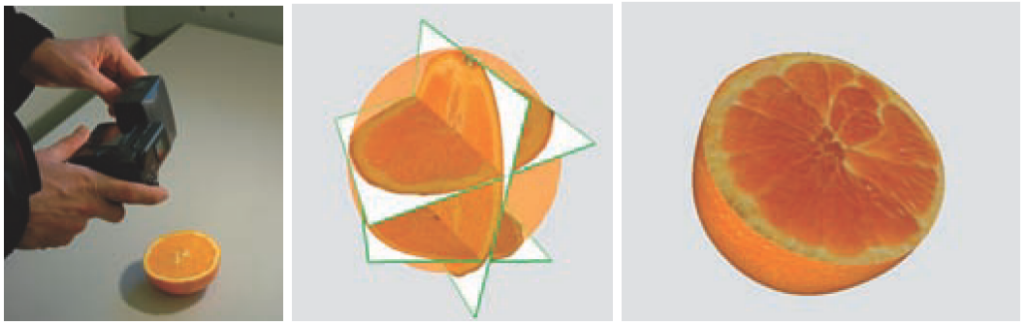


FIGURE 1.11 – Synthèse de texture par [POB+07]. L'utilisateur prend des photographies de la surface interne d'un fruit, qu'il va positionner à l'intérieur de l'objet 3D correspondant. Enfin, à l'aide de leur algorithme d'interpolation, les auteurs sont capables de représenter la texture interne de n'importe quel coupe du fruit.

tétraèdres sont plus compacts qu'une texture 3D haute résolution. Deuxièmement, étant donné le fait que chaque tétraèdre a sa propre texture, il est possible de réorganiser l'ensemble des tétraèdres du modèle pour simuler une texture solide dont l'apparence varie à l'intérieur de l'objet. Cependant, l'inconvénient principal de cette méthode est le fait que, pour que la texture ait une précision suffisante, la tétraédrisation de l'intérieur doit être assez fine, ce qui peut devenir problématique lorsqu'on utilise plusieurs objets en même temps ou plusieurs objets qui sont très volumineux. Cette méthode a permis de représenter la structure interne d'un kiwi et d'une pastèque (voir figure 1.12).

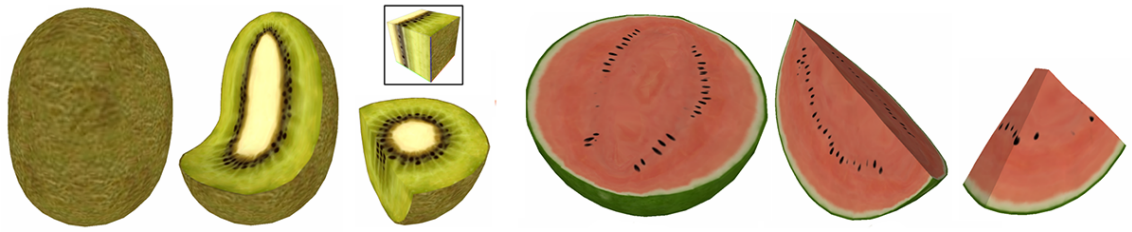


FIGURE 1.12 – Résultats obtenus par Takayama *et al.* [TOII08].

Toujours dans l'optique de représenter l'intérieur d'objets 3D de manière compacte, Takayama *et al.* présentent cette fois-ci les surfaces de diffusion [TSNI10], une extension des courbes de diffusion [OBW+08] appliquée aux volumes 3D, et qui permet de représenter des transitions douces entre les couleurs d'un objet 3D. Ce modèle de représentation consiste à définir un ensemble de surfaces 3D colorées qui décrivent la distribution de couleurs au sein du volume. Les couleurs de la structure interne de ce dernier sont ensuite obtenues en diffusant les couleurs de chaque côté de ces surfaces, et en les mélangeant aux couleurs des surfaces voisines. Grâce à cette méthode, les auteurs ont réussi à représenter la structure interne de plusieurs fruits différents, tout en étant capable de représenter les transitions douces qui existent entre les couleurs qu'on trouve au sein de vrais fruits (voir figure 1.13). Bien que cette méthode présente des résultats intéressants, elle est capable de ne représenter que des transitions de couleurs. En effet, le manque d'informations sur les textures n'a pas encore été traité par les auteurs, et la combinaison des courbes de diffusion à la synthèse de textures représenterait une direction intéressante à prendre pour des futures extensions.

FIGURE 1.13 – Résultats obtenus à l'aide des courbes de diffusion de Takayama *et al.* [TSNI10].

Wang *et al.* introduisent une représentation vectorielle et multi-échelle des textures solides dans [WYZG11]. Dans ce modèle de représentation, les objets sont représentés par des fonctions implicites utilisant des fonctions de distance signée. L'utilisateur peut ensuite créer des objets complexes en combinant ces fonctions implicites dans une structure arborescente. Grâce à cette méthode, il est ainsi possible de créer des volumes dont la structure interne complexe est composée d'un grand nombre de petites entités. L'avantage principal de cette méthode réside dans la structure arborescente des fonctions. En effet, cette dernière permet de décomposer une structure difficile à créer en plusieurs petites structures qui sont plus faciles à représenter. De plus grâce au fait que les éléments de cette structure sont des fonctions implicites, la représentation d'un objet complexe reste compacte, tout en étant capable d'être affichée à une résolution élevée. Bien que la modélisation de phénomènes géologiques reste le principal intérêt des auteurs (figure 1.14a), ces derniers ont également utilisé leur méthode pour modéliser la structure interne d'une orange (figure 1.14b).

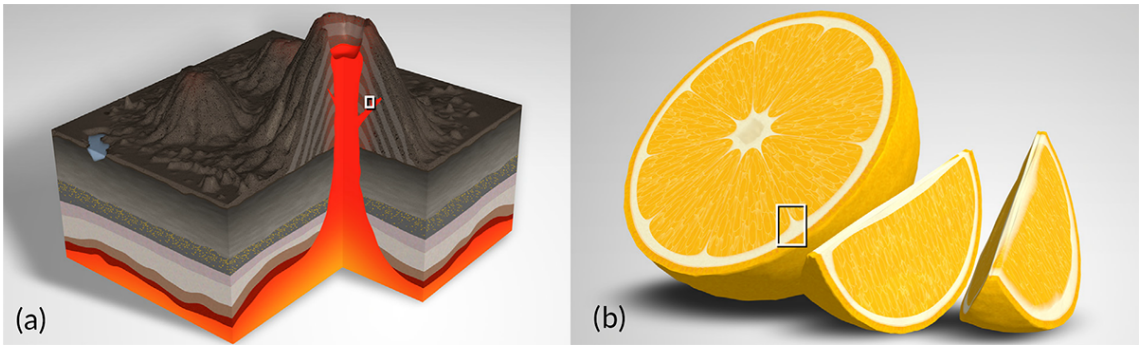


FIGURE 1.14 – Représentation vectorielle de la structure interne (a) d'un volcan et (b) d'une orange à l'aide de la méthode de Wang *et al.* [WYZG11].

1.2.2 Méthodes de représentation de défauts de fruits

1.2.2.1 Simulation basée image

Les techniques de simulation de défauts basées image consistent à faire l'acquisition et la mesure de photographies dans le but de les utiliser pour créer des imperfections dynamiques (c'est-à-dire qui évoluent au fil du temps).

Gu *et al.* introduisent une méthode qui permet de générer des textures dynamiques à l'aide de photographies d'objets réels [GTR+06]. Pour ce faire, les auteurs utilisent un

dispositif composé de multiples caméras et de multiples sources de lumière (figure 1.15b) pour récupérer une large base de données de photographies d'objets qui se détériorent au fil du temps. Parmi ces objets, les auteurs ont capturé l'évolution de la peau d'une pomme et d'une banane au fil du temps (figure 1.15a).

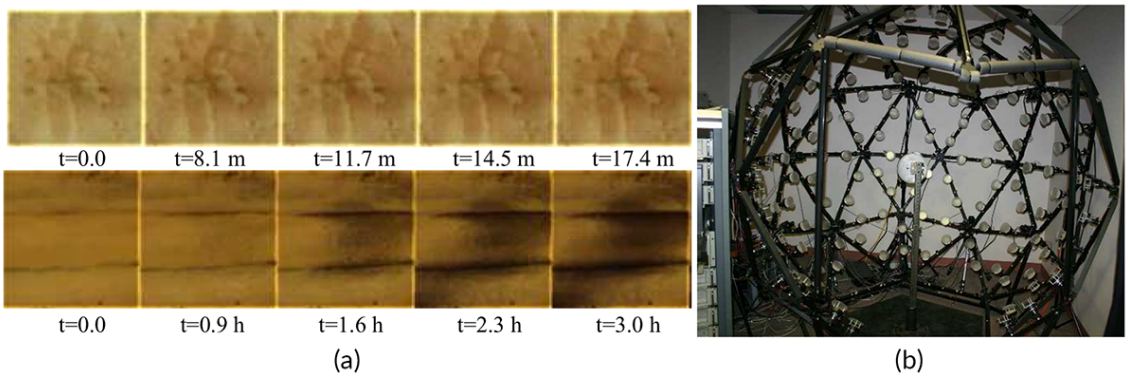


FIGURE 1.15 – Quelques échantillons des captures effectuées sur une pomme ((a) haut) et sur une banane ((a) bas) à l'aide du dispositif mis en place par Gu *et al.* [GTR+06] (b).

À l'issue de la phase de capture, les photographies ainsi obtenues permettent de calculer un modèle d'illumination de type BRDF qui va être utilisé pour texturer des objets 3D. Étant donné le fait que ces méthodes sont basées sur des photographies d'objets réels, elles garantissent des résultats réalistes, et permettent ainsi d'éviter de devoir créer des modèles physiques réalistes. Cependant, cette méthode présente deux défauts. Premièrement, les auteurs ne simulent que l'aspect visuel du vieillissement du fruit sans prendre en compte les facteurs biologiques qui interviennent lors du processus de vieillissement. Et deuxièmement, l'approche basée image n'est pas forcément pertinente pour représenter le vieillissement des fruits. En effet, le pourrissement d'un fruit dure en général plusieurs semaines, et le fait de devoir effectuer ce type de mesure sur une grande variété d'espèces de fruits peut s'avérer fastidieux et chronophage.

1.2.2.2 Simulation physiquement réaliste

Tout au long de son cycle de vie, l'apparence d'un fruit subit plusieurs changements (vieillissement, sénescence, maladies, etc...) qui dépendent de phénomènes chimiques, biologiques ou physiques. Alors qu'il est relativement aisé de représenter des fruits 3D d'aspect "propre" et "frais", la tâche devient difficile lorsqu'on essaye de simuler les différents facteurs de vieillissement qui altèrent leur apparence.

Les travaux de Kider *et al.* pour simuler la sénescence et le pourrissement du fruit [KRB11] sont très certainement les plus aboutis en informatique graphique. En effet, les auteurs sont les premiers à proposer une méthode qui lie la prolifération de bactéries et la transpiration du fruit au sein d'un même modèle. Ils présentent ainsi une approche physique basée sur les recherches effectuées en botanique pour simuler la sénescence et le pourrissement ayant lieu chez la pomme, la tomate et l'orange après la récolte.

Kider *et al.* prennent en compte trois facteurs de vieillissement dans leurs travaux. Les deux premiers sont la prolifération des champignons et de la pourriture molle à la surface

du fruit. Typiquement, un fruit attrape une maladie lorsqu'un point à sa surface est compromis, soit à cause d'une blessure ou soit à cause d'une vulnérabilité structurelle. Une fois que la pourriture molle attaque le fruit, sa surface s'affaiblit et rend ainsi possible la pénétration de champignons. Ces derniers vont alors se répandre aux endroits où la pourriture molle aura infecté le fruit, mais aussi aux endroits où la concentration en nutriments est suffisante. En effet, au fil du temps, les champignons vont attaquer les nutriments et faire diminuer leur concentration, causant ainsi un ralentissement de leur propre prolifération.

Pour représenter ce phénomène, les auteurs utilisent un modèle de réaction-diffusion qui va agir sur trois textures différentes (voir figure 1.16). La première texture représente la distribution des nutriments le long de la surface du fruit et sert à orienter, accélérer ou ralentir la croissance fongique. La deuxième texture utilisée sert quant à elle à traquer les zones où il y a formation de colonies fongiques. Enfin, la texture de pourriture molle représente les zones de croissance bactérienne qui se forment sur le fruit, et sert de carte de probabilité pour la propagation de nouvelles colonies fongiques. La figure 1.16 illustre un exemple de simulation sur ces trois textures dans le cas d'une pomme. La couleur d'un pixel sur une texture varie du noir (pour une densité minimale égale à 0) au blanc (pour une densité maximale égale à 1), et nous pouvons constater sur la figure 1.16 que la texture de nutriments s'assombrit au fil du temps alors que les textures de bactéries et de pourriture molle s'éclaircissent.

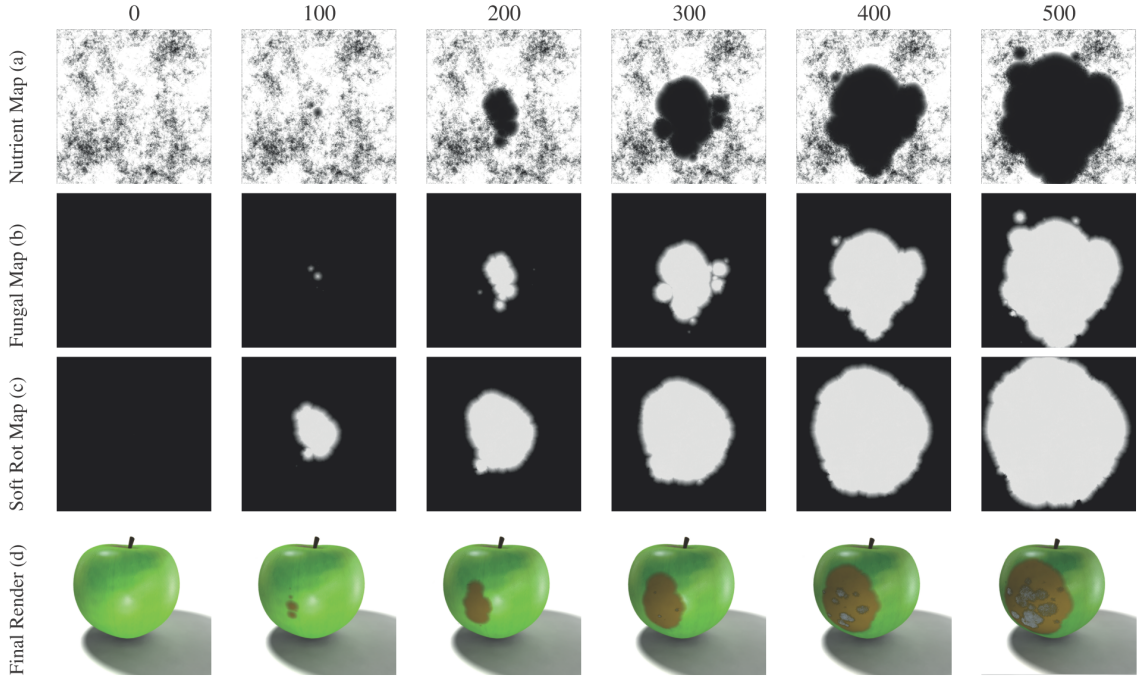


FIGURE 1.16 – Résultats obtenus par [KRB11] : (a) diminution de la concentration en nutriments (b) croissance des colonies fongiques (c) propagation de la pourriture molle et (d) rendus finaux.

Le troisième facteur de vieillissement qui est pris en compte par [KRB11] est la transpiration du fruit. Comme tous les êtres vivants, un fruit est composé d'eau, et la teneur en eau varie d'une espèce à l'autre. Cependant, un fruit va transpirer tout au long de son cycle de vie, entraînant ainsi une perte d'eau qui va provoquer la diminution de sa masse et le rétrécissement de son volume. Pour simuler ce phénomène d'affaissement, les au-

teurs représentent le volume interne du fruit par un objet déformable et sa peau par un tissu relié aux sommets de ce volume. Au fur et à mesure que la simulation progresse, un système de masse-ressort va alors calculer les forces appliquées sur le fruit en fonction de son taux de transpiration, faisant ainsi changer la forme et le volume de son mésocarpe. La figure 1.17 illustre le phénomène de transpiration sur un modèle 3D de tomate et un modèle 3D d'orange.

FIGURE 1.17 – Résultats obtenus par [KRB11] : simulation de l'affaissement (haut) d'une tomate et (bas) d'une orange.

Les résultats de [KRB11] sont très convaincants et permettent d'obtenir des résultats physiquement réalistes. Nous noterons cependant quelques défauts. Premièrement, la réaction-diffusion ne s'applique qu'à la surface du fruit. Pour un réalisme accru, une extension pourrait consister par exemple à distribuer les champignons, la pourriture molle et les nutriments dans tous le volume du fruit et non pas qu'à sa surface. Une deuxième limitation réside dans le fait que les défauts qui sont pris en compte n'interviennent qu'après la récolte. Il serait ainsi intéressant de proposer une méthode permettant de représenter les défauts qui apparaissent avant la récolte, et pendant la phase de croissance du fruit, comme les crevasses par exemple.

Dans le but d'accroître le réalisme du modèle de transpiration de [KRB11], Liu *et al.* proposent une extension de celui-ci [LCW+12] qui utilise un maillage tétraédrique à la place d'un maillage surfacique, pour représenter le volume interne du fruit. Contrairement à un maillage surfacique, un maillage tétraédrique est capable de représenter les différentes couches du péricarpe, ce qui rend possible la simulation du flétrissement sur des coupes de fruits (figure 1.18a). De plus, ce type de maillage permet aux auteurs de calculer la dynamique des couches interne en utilisant la méthode des éléments finis sur les sommets, ce qui donne des résultats physiquement plus corrects que ceux de [KRB11]. Dans [LYC+15], les auteurs améliorent leur modèle pour pouvoir représenter des drupes complètement déshydratées (figures 1.18b et 1.18c). Ainsi, toujours à l'aide de la méthode des éléments finis, ils sont capables de représenter des prunes séchées, des jujubes séchées (ou dattes chinoises) ainsi que des raisins secs. Cependant, les deux principaux défauts de ce type de méthode résident dans la contrôlabilité du résultat et le paramétrage de la simulation. En effet, les méthodes de simulation physiquement réalistes se basent sur des

processus biologiques qui sont en général difficiles à modéliser, et le modèle mathématique qui en résulte est souvent composé de paramètres difficile à appréhender pour une personne qui ne s'est pas suffisamment documenté sur la question.

FIGURE 1.18 – Résultats obtenus par Liu *et al.* : (a) une coupe de pomme déshydratée, (b) une prune déshydratée et (c) un jujube déshydraté.

1.2.2.3 Simulation visuellement réaliste (ou simulation empirique)

Comme nous l'avons expliqué dans la section 1.1.5, il existe une énorme quantité de défauts possibles, car chaque fruit présente des défauts qui sont propres à son espèce. Cette large variété d'imperfections possibles pose un problème non négligeable aux méthodes qui visent une simulation physiquement réaliste. En effet, les modèles physiquement réalistes et physiquement corrects sont difficiles à créer à cause notamment du manque de littérature existant dans le domaine. De plus, à cause de la difficulté que cela représente et du temps qu'il faut pour les mettre en place, ce type de méthode ne représente en général que quelques aspects du vieillissement.

C'est pour répondre à cette limitation que certains travaux introduisent des méthodes de simulation visuellement réalistes. Contrairement aux méthodes physiquement réalistes, qui tentent de comprendre et de reproduire des phénomènes biologiques complexes, les simulations visuellement réalistes créent des modèles empiriques basés sur des observations de ces phénomènes, et dont le seul but est de générer des résultats qui paraissent suffisamment réels.

Les premiers travaux sur le vieillissement empirique remontent à l'époque où Blinn introduisait le bump mapping pour créer des rides sur une fraise ou une orange dans [Bli78]. Cette technique très simple à mettre en place est facilement contrôlable et peut donner des résultats suffisamment réalistes (figure 1.19). Depuis, un grand nombre de travaux ont été effectués sur la simulation de rides [NP15, WKMMT99], et plus généralement sur la représentation des tissus, notamment parce qu'il s'agit d'une matière qui est omniprésente dans la plupart des productions 3D.

Dans le but de pouvoir représenter une large gamme de défauts à la surface des fruits, Verhulst *et al.* introduisent une méthode basée sur l'utilisation de systèmes de particules [VNM15]. Selon les auteurs, l'apparence complexe du vieillissement d'un fruit peut être séparée en plusieurs caractéristiques visuelles "simples". En observant une banane en phase de sénescence par exemple, les auteurs constatent que sa peau vieillissante est en fait

FIGURE 1.19 – Vieillissement d'une pomme à l'aide du bump mapping de Blinn [Bli78] : un fruit frais (gauche) et le même fruit vieilli avec des rides (droite).

composée de trois motifs qui se répètent le long de sa surface (figure 1.20b) : des points noirs, des rayures noires et des régions noires de formes non spécifiques. Pour représenter ces caractéristiques visuelles, les auteurs assignent chacune d'entre elles à un système de particules qui se chargera de générer une texture reproduisant leur motif. Pour chaque système de particules, l'utilisateur doit définir : le moment où la propagation débute, la rapidité de propagation, la densité de propagation, les régions dans lesquelles elle a lieu, et sa durée. Une fois cette phase de paramétrage terminée, une simulation de propagation est effectuée pour chaque système de particules, générant ainsi autant de textures qu'il y a de systèmes de particules. Enfin, à l'issue de cette phase de propagation, le logiciel combine ces textures en une seule et même texture qui va être plaquée sur le modèle final pour représenter son vieillissement (figure 1.20c). Le fonctionnement de leur méthode est illustré sur la figure 1.20a.

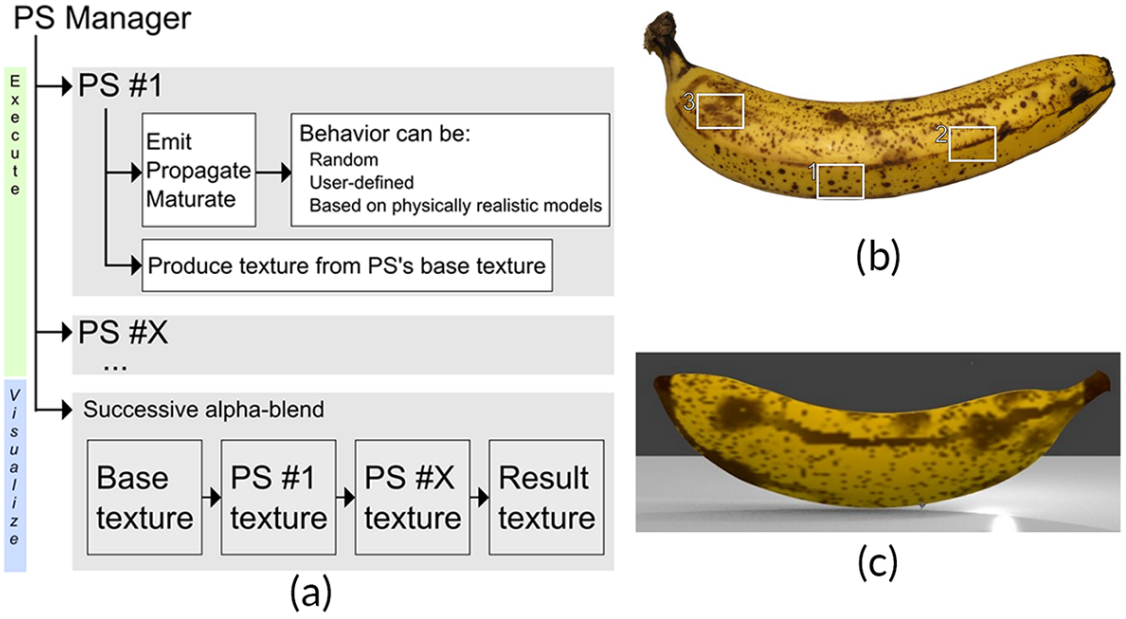


FIGURE 1.20 – Méthode de Verhulst *et al.* [VNM15] : (a) Schéma de fonctionnement de la méthode, (b) photo d'une vrai banane avec mise en évidence des motifs (1 : points noirs, 2 : rayures noires, 3 : régions noires de formes non spécifiques), et (c) résultat de la simulation.

La contrôlabilité est certainement la principale force de [VNM15]. En effet, en séparant l'apparence en plusieurs motifs simples, l'utilisateur est ainsi capable de composer celle-ci comme il le souhaite. Cependant, bien que les résultats sur la surface du fruit soient visuellement réalistes, leur méthode ne prend pas en compte le rétrécissement du volume dû à la transpiration, contrairement à [KRB11]. Ainsi, si nous continuons par exemple la simulation sur le modèle de la figure 1.20c, nous nous retrouverons avec une apparence de la surface du fruit qui ne concorde pas avec sa forme générale. Une extension intéressante de cette méthode consisterait alors à prendre en compte l'influence qu'a le pourrissement sur la forme générale du fruit, afin de pouvoir générer des modèles plus réalistes.

1.3 Méthodes de modélisation procédurale

La modélisation procédurale s'est révélée très utile dans un grand nombre de domaines spécifiques à l'informatique graphique [EMP+02]. Elle permet de générer des objets 3D de manière automatique à l'aide de procédures, de variables et de fonctions mathématiques. De plus, pour ajouter de la variété au résultat final, l'aléatoire est souvent utilisé dans ce type de méthode. Ainsi, le simple fait de créer le modèle procédural d'un arbre peut nous permettre de générer une forêt toute entière dans laquelle les arbres sont tous différents les uns des autres.

Ce type de technique est souvent utilisé pour générer des objets qui sont trop complexes à modéliser de manière interactive, c'est-à-dire à l'aide d'un logiciel de modélisation 3D classique. Ainsi, si l'utilisateur a besoin de modéliser une ville toute entière par exemple, plutôt que d'essayer de modéliser chacun des immeubles à la main, il va pouvoir utiliser le modèle procédurale d'un bâtiment. Celui-ci va alors générer de manière automatique une large gamme d'immeubles 3D différents qui vont peupler la ville virtuelle, faisant ainsi gagner énormément de temps sur le processus de modélisation.

Les travaux que nous présentons dans ce mémoire sont basés sur une technique de modélisation procédurale. Avant de présenter cette technique en détail dans la section suivante, nous présentons ici un état de l'art des méthodes procédurales divisé en trois parties : la modélisation d'éléments urbains, la modélisation d'éléments naturels et la modélisation procédurale guidée.

1.3.1 Modélisation procédurale d'éléments urbains

Les méthodes de modélisation d'immeubles font partie des méthodes procédurales les plus développées. La plupart des méthodes de cette catégorie utilisent une variante des L-systèmes (présentés dans la section suivante) pour générer des immeubles 3D sur la surface un terrain 2D. Ces méthodes présentent des résultats très convaincants, mais demandent énormément d'intervention de la part de l'utilisateur.

Bao *et al.* [BSW13] proposent une méthode qui, à l'aide d'une façade d'immeuble prise en entrée, permet de générer plusieurs variations de celle-ci, tout en conservant l'essence de son design. Parish *et al.* proposent quant à eux dans [PM01] un modèle permettant de modéliser des villes entières. Pour ce faire, les auteurs commencent par créer un plan au sol, puis appliquent un L-système qui va faire "croître" les immeubles sur ce

plan. Dans [MWH+06], Müller *et al.* présentent *Computer Generated Architecture Shape* (ou CGA Shape), une méthode permettant de générer un grand nombre d'immeubles 3D de formes différentes à l'aide d'une seule et même grammaire. Pour ce faire, la méthode crée dans un premier temps un volume extrudé à partir d'un "terrain", et qui est composé de plusieurs étages. Ensuite, tout en respectant des règles qui ont été spécifiées dans une grammaire, les façades de ce volume extrudé sont subdivisées en murs, fenêtres et portes. Ces règles sont très puissantes et peuvent par exemple ajouter des contraintes à la génération de certains types d'éléments, comme par exemple : "une porte d'entrée ne peut être créée que si nous nous trouvons au rez-de-chaussée ou sur une terrasse". De plus, pour ajouter de la variété aux résultats obtenus, l'utilisateur peut aussi ajouter des paramètres ou de l'aléatoire à ses règles. Les grammaires de Müller *et al.* représentent le modèle le plus développé pour la génération d'immeubles 3D. La figure 1.22 montre le genre de résultats qu'il est possible d'obtenir à l'aide d'une seule et même grammaire.

FIGURE 1.21 – Ensemble d'immeubles 3D obtenus à l'aide d'une seule grammaire de CGA Shape [MWH+06].

1.3.2 Modélisation procédurale d'éléments naturels

Les objets naturels tels que les arbres, les fleurs ou les paysages naturels sont d'autres types d'objets trop complexes à construire par modélisation interactive. Ainsi, la plupart du temps, un artiste ne cherchera pas à modéliser chaque feuille une à une pour créer un arbre, mais utilisera plutôt un modèle procédural, comme les L-systèmes par exemple (présentés dans la section suivante), pour générer l'ensemble des feuilles qui vont peupler le "squelette" de cet arbre.

Runions *et al.* [RFL+05] présentent plusieurs algorithmes biologiques permettant de générer des feuilles ainsi que leurs nervures. Pour ce faire, les auteurs simulent les interactions qui interviennent entre trois processus : la croissance de la feuille, la distribution spatiale des sources d'auxine (hormone de croissance) et le développement des veines. Dans [RLP07], les auteurs étendent leur méthode à la troisième dimension pour pouvoir modéliser des arbres. Ils utilisent ainsi leur algorithme de colonisation pour simuler la compétition pour l'espace et la lumière qui existe entre les branches d'un arbre lors de sa croissance. Lintermann *et al.* [LD99] ont développé quant à eux une méthode interactive permettant de modéliser des plantes et des arbres. Ces travaux donneront naissance à *Xfrog*, un modeleur 3D organique et procédural utilisé dans beaucoup de productions cinématographiques importantes. Ce modeleur est capable de créer et d'animer des arbres, des fleurs ainsi que des formes architecturales.

Benes *et al.* présentent une méthode dans [BCS03] qui permet de simuler le développement et l'entretien de jardins par des agents virtuels autonomes. Pour ce faire, un jardin

"sauvage" composé d'herbes et de plantes est généré dans un premier temps. Ensuite l'utilisateur définit la distribution des agents virtuels dans le jardin ainsi que les tâches qui sont assignées à chacun d'eux. Un agent peut par exemple planter des graines, arroser des plantes, ou créer des chemins à l'aide de pavages. Une fois que les paramètres sont définis, l'utilisateur lance alors le système qui va se charger de simuler le développement de cet écosystème pour obtenir la structure du jardin final (voir figure 1.22)

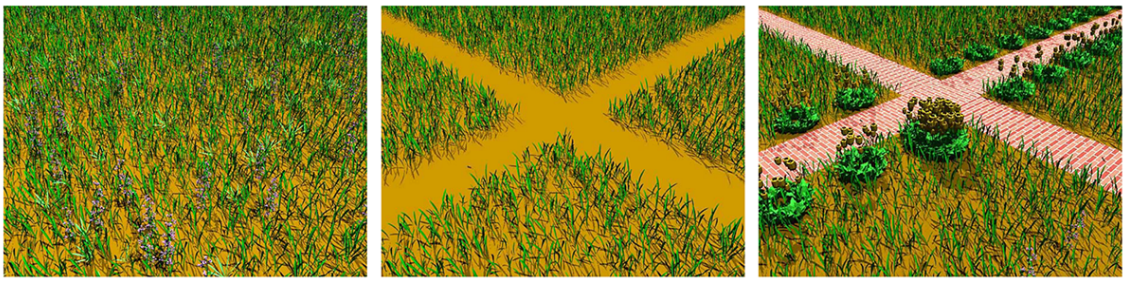


FIGURE 1.22 – Résultats de [BCS03] : (de gauche à droite) un jardin virtuel avant, pendant et après le passage des agents virtuels.

1.3.3 Modélisation procédurale guidée

Bien que les méthodes de modélisation procédurale permettent de représenter des objets complexes, une grande majorité d'entre elles présente un défaut majeur. En effet, lorsqu'on utilise ce type de méthode pour faire de la modélisation, on est souvent confronté au manque de contrôle et à la faible prédictibilité du résultat final. Benes *et al.* tentent d'apporter une solution à ce problème en introduisant dans [BSMM11] le concept de modélisation procédurale guidée. Cette approche consiste à découper un système qui génère un modèle 3D complexe, en plusieurs petits systèmes générant chacun une partie simple de ce modèle (figure 1.23). Ces petits systèmes sont appelés guides et sont capables de communiquer entre eux, ce qui facilite ainsi le processus de modélisation. Ce dernier ne se fait donc plus en décrivant l'objet à modéliser dans son intégralité, mais en décrivant des petites régions qui le composent.

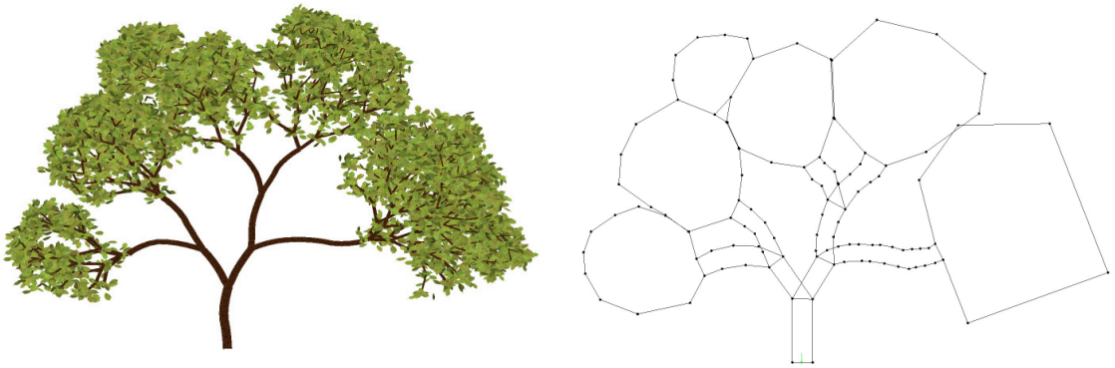


FIGURE 1.23 – Résultats de [BSMM11] : (gauche) Le modèle 3D d'un arbre et (droite) le partitionnement de ce modèle en guides.

1.4 Modèle de croissance

Pour répondre à notre problématique, nous avons choisi d'utiliser lors de nos travaux un modèle de croissance basé sur les 3G-cartes L-systèmes. Ce concept, introduit par Terraz *et al.* [TGM+09], est une extension des systèmes de Lindenmayer (ou L-systèmes)[Lin68] qui est basée sur l'utilisation des cartes généralisées de dimension trois (3G-cartes) [Lie91, Lie94]. Avant de présenter en détails ce concept, nous rappellerons dans un premier temps certaines notions importantes sur les cartes généralisées de dimension trois et sur les systèmes de Lindenmayer. Finalement, dans la seconde partie de cette section, nous verrons en quoi consistent les 3G-cartes L-systèmes et quels sont les avantages de l'utilisation de ce concept pour la réalisation de nos travaux.

1.4.1 3G-cartes

Les cartes généralisées de dimension trois (ou 3G-cartes) font partie des modèles de représentation par frontières (ou BRep pour *Boundary Representation*). L'espace y est subdivisé en cellules de différentes dimensions, avec la dimension 0 qui correspond aux sommets, la dimension 1 aux arêtes, la dimension 2 aux faces et la dimension 3 aux volumes. Ce concept, qui a été introduit par Pascal Lienhardt [Lie94], est un modèle qui permet de représenter la topologie de quasi-variétés cellulaires quelconques de dimension trois, orientables ou non, avec ou sans bord. Une quasi-variété de dimension trois est, de manière intuitive, un objet 3D que l'on peut obtenir par assemblage de cellules de dimension 3 (des volumes) et ce, uniquement le long de cellules de dimension 2 (des faces). Nous allons maintenant définir les notions importantes nécessaires à la compréhension de ce mémoire.

Définition. Une carte généralisée de dimension trois, ou 3G-carte, est définie par un 5-uplet $(B, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$ tel que :

- B est un ensemble fini de brins ;
- α_0, α_1, et α_2 sont des involutions sans point fixe ;
- α_3 est une involution avec ou sans point fixe ;
- $\alpha_0\alpha_2$ et $\alpha_1\alpha_3$ sont des involutions.

Brin. Le modèle de 3G-carte utilise un unique type d'élément : le brin. Intuitivement, ce dernier peut être vu comme un sommet, et être graphiquement représenté par une demi-arête (voir figure 1.24).

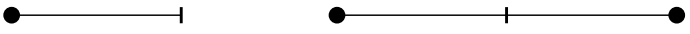


FIGURE 1.24 – représentation d'un brin (gauche) et d'une arête composée de deux brins (droite).

Involutions et brins libres. Pour une 3G-carte, α_i est une involution si pour $0 \leq i \leq 3$ et pour un brin b on a : $\alpha_i(\alpha_i(b)) = b$. De plus, une involution α_i contient un point fixe si pour un brin b on a : $\alpha_i(b) = b$. Et enfin, un brin est dit libre par α_i si il est un point fixe

pour l'involution α_i. Sur la figure 1.25, on peut voir par exemple que le brin $b1$ est libre par α_1 et que α_0 est une involution pour $b2$.

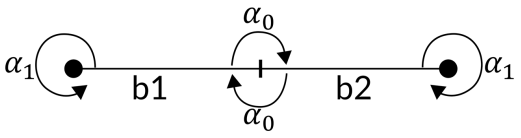


FIGURE 1.25 – exemple d'involutions au sein d'une arête formée par deux brins.

Opérateurs d'assemblage. Les involutions $\alpha_i (0 \leq i \leq 3)$ peuvent être vues comme des opérateurs d'assemblage permettant d'établir des relations entre les brins d'une carte (voir figure 1.26). Ils peuvent ainsi connecter des brins entre eux, et sont définis comme suit :

- α_0 permet de relier deux brins entre eux pour former une arête ;
- α_1 permet de coudre deux arêtes entre elles ;
- α_2 permet de relier deux faces entre elles le long des arêtes de leurs bords ;
- α_3 sert à assembler des volumes entre eux le long des faces de leurs bords pour former des structures de volumes.

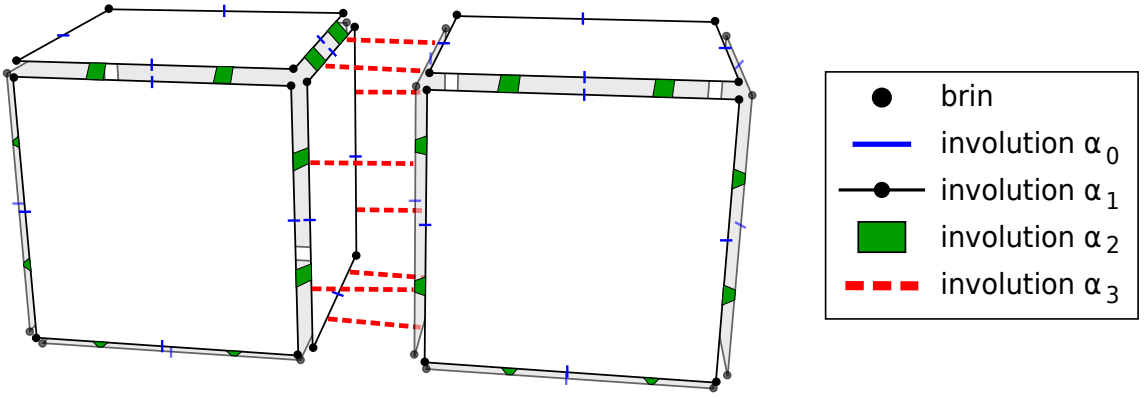


FIGURE 1.26 – Vue éclatée de deux cubes adjacents mettant en évidence les différents opérateurs d'assemblage.

À l'aide des informations que nous avons présenté ci-dessus, nous pouvons déduire qu'il est possible de représenter n'importe quelle structure volumique à partir d'un ensemble de brins munis des opérateurs d'assemblage adéquats. De plus, en se servant des opérateurs d'assemblage, il est possible de récupérer à partir d'un brin donné, un sous ensemble de brins de la structure à laquelle appartient celui-ci. Pour ce faire, il suffit de choisir la suite d'involutions qui vont être utilisées pour parcourir la structure depuis le brin d'origine. On appelle cette succession d'involutions une orbite.

Orbite. L'orbite d'un brin b par rapport à un ensemble d'involutions $\alpha_i, \ldots, \alpha_j$ est définie comme l'ensemble des brins que l'on peut obtenir à partir de b par application de $\alpha_i, \ldots, \alpha_j$ dans un ordre quelconque. On note cette orbite $< \alpha_i, \ldots, \alpha_j > (b)$. Par exemple sur la figure 1.27, on a : $< \alpha_0, \alpha_1 > (b1) = \{b1, b2, b6, b4\}$.

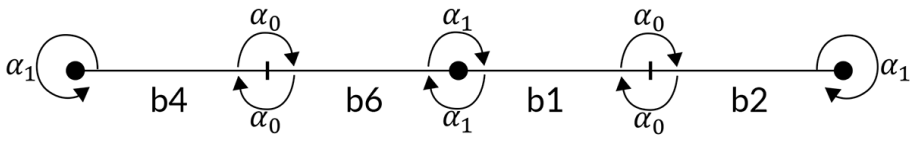


FIGURE 1.27 – Exemples d'involutions α_0 et α_1 dans une structure de brins.

Composante connexe. L'orbite d'un brin b peut par exemple être utile lorsqu'on veut déterminer la composante connexe d'une 3G-Carte $(B, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$. Une composante connexe est définie par l'ensemble des brins qu'on obtient à partir de l'un d'eux par application de l'orbite $< \alpha_0, \alpha_1, \alpha_2, \alpha_3 > (b)$.

Cellule. Pour $0 \leq i \leq 3$, on définit une i-cellule incidente à un brin b par l'orbite utilisant toutes les involutions sauf α_i. Les i-cellules d'une 3G-carte sont :

- La 0-cellule, qui représente l'ensemble des brins du sommet incident à b, est définie par $< \alpha_1, \alpha_2, \alpha_3 > (b)$;
- La 1-cellule, qui représente l'ensemble des brins de l'arête incidente à b, est définie par $< \alpha_0, \alpha_2, \alpha_3 > (b)$;
- La 2-cellule, qui représente l'ensemble des brins de la face incidente à b, est définie par $< \alpha_0, \alpha_1, \alpha_3 > (b)$;
- La 3-cellule, qui représente l'ensemble des brins du volume incident à b, est défini par $< \alpha_0, \alpha_1, \alpha_2 > (b)$;

Notion de bord. Le bord d'une 3G-carte est constitué par l'ensemble des brins étant des points fixes pour l'involution α_3. Ainsi, pour une 3G-carte $(B, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$, son bord est défini par la 2G-carte $(B', \alpha'_0, \alpha'_1, \alpha'_2)$ avec :

- B' qui est un sous-ensemble de B réduit aux brins du bord (tel que $\alpha_3(b) = b$) ;
- α'_0 et α'_1 qui sont des restrictions de α_0 et α_1 aux brins du bord ;
- α'_2 qui est la restriction de $\alpha_2\alpha_3$ aux brins du bord.

1.4.2 L-systèmes

Le concept de L-systèmes (ou systèmes de Lindenmayer) est un formalisme mathématique introduit par Aristid Lindenmayer en 1968 [Lin68] comme le fondement d'une théorie axiomatique du développement des plantes. Ce langage formel est basé sur la notion de réécriture, où l'idée de base est de définir un objet complexe en remplaçant successivement les parties d'un objet simple, grâce à un ensemble de règles qu'on appelle règles de réécriture (ou règles de production). Ces règles vont ainsi permettre, à chaque nouvelle étape du processus d'évolution, de mettre le système dans un état plus évolué en agissant sur des symboles. Ces derniers représentent les cellules de l'organisme qu'on souhaite modéliser (exemple : une feuille) et, à chaque pas d'exécution du système, ces symboles vont ainsi être assemblés entre eux pour former des mots qui vont représenter la structure de l'objet modélisé.

On remarquera qu'il existe une grande similarité entre les L-systèmes et le travail de Noam Chomsky sur les grammaires formelles [Cho57]. La principale différence réside cependant dans la manière d'appliquer les règles de production. En effet lorsqu'on utilise les grammaires de Chomsky, les règles sont appliquées de manière séquentielle alors que dans le cas des L-systèmes les règles sont appliquées en parallèle, remplaçant ainsi toutes les lettres simultanément. Cette différence reflète le côté biologique des L-systèmes, ainsi que leur utilité dans la représentation de la division cellulaire des organismes multicellulaires, au sein desquels plusieurs divisions ont lieu en même temps.

1.4.2.1 Définition et Exemple

Définition. Un L-système est défini par un 4-uplet (V, S, ω, P) où :

- V est l'alphabet du système qui contient l'ensemble des symboles ;
- S représente les valeurs constantes du système, aussi appelées symboles particuliers ;
- $\omega \in V^+$ est l'axiome de départ choisi parmi l'ensemble des mots pouvant être construits à partir de l'alphabet du système. Il représente l'état initial du système ;
- P est l'ensemble des règles de réécriture des symboles de l'alphabet V.

Pour mieux comprendre comment fonctionnent les L-systèmes, prenons l'exemple du tout premier L-système de Lindenmayer permettant de représenter le développement d'une algue. On considère des mots formés de deux lettres A et B qui peuvent se trouver autant de fois qu'on veut dans un mot. Chaque lettre est associée à une règle de réécriture, et la grammaire est définie comme suit :

- $V = A, B$
- $S = \emptyset$
- $\omega = B$
- $P = (A \to AB) \wedge (B \to A)$

On peut voir dans l'exemple ci-dessus que l'ensemble P est composé de deux règles de réécriture. La première $(A \to AB)$ signifie qu'à chaque étape de l'exécution du système, chaque lettre A est remplacée par le mot AB. La seconde $(B \to A)$ quant à elle signifie que chaque lettre B est remplacée par la lettre A à chaque étape de l'exécution. Ainsi, lorsqu'on dérive ce L-système, on obtient les résultats suivants pour les cinq premières générations (ou dérivations) :

- Génération 0 : B.
- Génération 1 : A.
- Génération 2 : AB.
- Génération 3 : ABA.
- Génération 4 : $ABAAB$.
- Génération 5 : $ABAABABA$

Nous pouvons voir que le résultat qu'on obtient après les cinq premiers pas de dérivation du L-système est le mot "ABAABABA". Tel qu'il est présenté, ce mot possède une signification abstraite qui n'est pas facile à interpréter tel quel. Pour mieux le comprendre, il est donc nécessaire d'utiliser une méthode d'interprétation graphique.

1.4.2.2 Interprétation graphique des L-systèmes

Le concept de tortue LOGO a été introduit dans les années 60 par Seymour Papert, un chercheur du MIT. Cette méthode permet d'interpréter graphiquement les résultats obtenus par la dérivation d'un L-système.

La tortue est une entité qui se déplace selon trois axes qui lui sont propres : $\vec{H}$ pour sa direction, $\vec{L}$ pour la direction de la gauche et $\vec{U}$ pour la direction du haut. À chaque pas d'exécution du L-système, celle-ci reçoit des instructions de commande relatives à son propre repère et non pas à un référentiel fixe. Les opérations que la tortue peut exécuter sont : la rotation autour d'un de ses axes et le déplacement le long de l'un de ses axes. Ces opérations sont représentées par des symboles (voir figure 1.28) qu'on ajoute à la grammaire des L-systèmes :

- $F(s)$ avancer de s unités de longueur en traçant un segment ;
- $f(s)$ avancer de s unités de longueur sans tracer de segment.
- $+(\theta)$ tourner à gauche d'un angle θ autour de $\vec{U}$;
- $-(\theta)$ tourner à droite d'un angle θ autour de $\vec{U}$;
- $\wedge(\theta)$ tourner à gauche d'un angle θ autour de $\vec{L}$;
- $\&(\theta)$ tourner à droite d'un angle θ autour de $\vec{L}$;
- $/(\theta)$ tourner à gauche d'un angle θ autour de $\vec{H}$;
- $\backslash(\theta)$ tourner à droite d'un angle θ autour de $\vec{H}$.

En plus de ces symboles, deux opérateurs sont introduits pour permettre de représenter les structures arborescentes :

- [sert à empiler l'état de la tortue ;
-] dépile l'état depuis la pile d'états et l'affecte à l'état de la tortue.

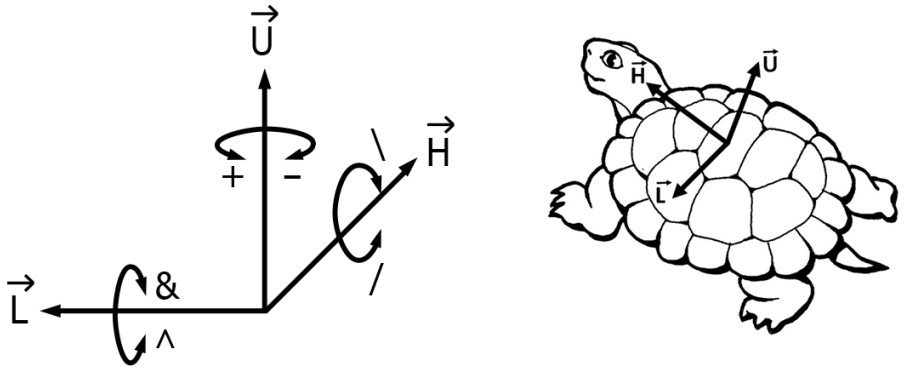


FIGURE 1.28 – Les opérations de la tortue LOGO dans un espace de dimension 3.

Pour mieux comprendre la puissance d'un tel ajout à la grammaire des L-systèmes, nous allons présenter ci-dessous l'exemple du L-système du triangle de Sierpinski, qui est défini comme suit :

- $V = A, B$
- $S = +, -$
- $\omega = A$
- $P = (A \rightarrow B - A - B) \wedge (B \rightarrow A + B + A)$
- $angle\ de\ rotation\ \theta = 60°$

Dans cet exemple, A et B veulent tous les deux dire "avancer en traçant un segment". Le symbole '$+$' quant à lui signifie "tourner à gauche d'un angle θ", et '$-$' veut dire "tourner à droite d'un angle θ". La figure 1.29 illustre quelques résultats de la dérivation de ce L-système.

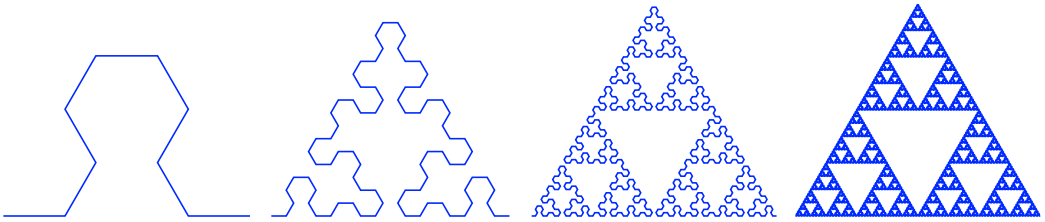


FIGURE 1.29 – Résultats obtenus avec le L-système du triangle de Sierpinski après deux, quatre, six et huit dérivations.

1.4.2.3 L-systèmes paramétriques

Dans le but de pouvoir contrôler plus finement l'application des règles de production, Prusinkiewicz *et al.* ont introduit les L-systèmes paramétriques [Pru04]. Par rapport aux L-systèmes classiques, les auteurs y ont ajouté la possibilité d'introduire des paramètres tels que la longueur des segments, leur épaisseur et bien d'autres. Voici un exemple de L-système paramétrique, dont le résultat est illustré sur la figure 1.30 :

- $\omega : !(3)F(1,1)$
- $p1 : F(s,t) \rightarrow t == 1 \rightarrow F(s,2)[-!(1)F(s,1)][+!(1)F(s,1)]F(s,2)!(1)F(s,1)$
- $p2 : F(s,t) \rightarrow t == 2 \rightarrow F(2 \times s, 2)$
- $p3 : !(e) \rightarrow e < 2 \rightarrow !(3)$

Le L-système paramétrique présenté ci-dessus présente un nouvel opérateur $!(e)$, qui permet de fixer l'épaisseur du trait à e. De plus, le module F a deux paramètres, s et t, au lieu d'un. Ici, s définie la longueur de la ligne à tracer et t indique si le segment tracé est un noeud ($t == 1$) ou un internoeud ($t == 2$).

Lors de la dérivation du L-système, la règle $p1$ est appliquée pour tous les segments F ayant le statut de noeud ($t == 1$), alors que la règle $p2$ est appliquée pour tous ceux qui ont le statut d'internoeud ($t == 2$). Quant à la règle $p3$, celle-ci est appliquée à chaque dérivation sur tous les segments. Ainsi, si l'épaisseur d'un segment est inférieure à 2, alors on incrémente son épaisseur.

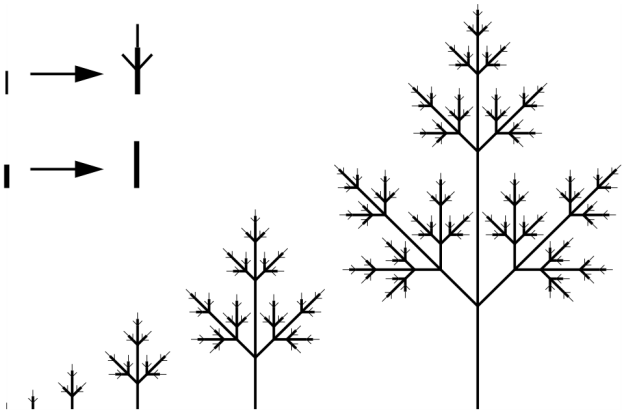


FIGURE 1.30 – Exemple de résultat obtenu à l'aide d'un L-système paramétrique.

1.4.2.4 L-systèmes dépendants du contexte

Dans certains cas, il peut être très intéressant de pouvoir appliquer une règle de production uniquement si certaines conditions sont réunies. Pour modéliser un arbre, une condition peut par exemple faire en sorte que ses branches ne peuvent pas pousser de chaque côté d'un noeud. Pour ce faire, il est possible d'utiliser des L-systèmes dépendants du contexte. Nous pouvons en distinguer deux types :

- 2L-Systèmes : ici, l'application des règles dépend du contexte gauche ou droit de l'ensemble des prédécesseurs ;
- 1L-Système : dans ce type de système, l'application des règles dépend soit du contexte gauche, soit du contexte droit de l'ensemble des prédécesseurs.

Les règles de production qui sont utilisées au sein des L-systèmes dépendants du contexte sont de la forme : $A_g < A > A_d \rightarrow M$. Dans cet exemple, le mot A sera changé en M, si et seulement si il est entouré des mots A_g et A_d, qui sont respectivement le contexte gauche et le contexte droit de A. La figure 1.31 montre le résultat de l'application du L-système suivant, qui permet de simuler la propagation des segments I dans une structure arborescente à partir de sa racine :

- $\omega :\ I[J]J[J]J[J]J$
- $p1 :\ I < J \rightarrow I$

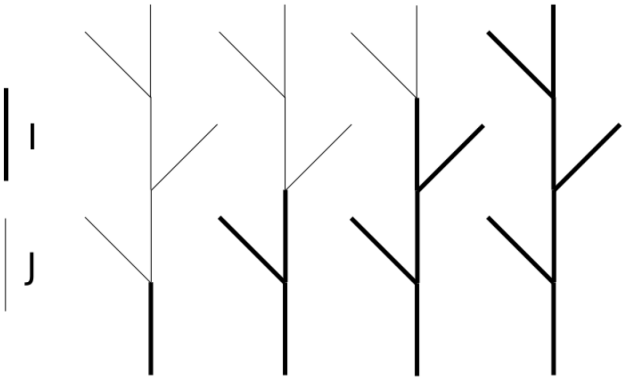


FIGURE 1.31 – L-système dépendant du contexte. Propagation des segments I dans la structure arborescente.

1.4.2.5 Maps et Cellworks L-systèmes

Les L-systèmes présentés jusqu'à présent s'avèrent être très adaptés à la modélisation de la croissance d'organismes divers. Cependant, bien qu'ils soient particulièrement adaptés aux structures "linéaires", les L-systèmes sont inappropriés pour manipuler des structures topologiques à deux ou trois dimensions. C'est en partant de cette observation que deux extensions des L-systèmes ont été présentées par Prusinkiewicz [PL96] et Lindenmayer [LR78] : les maps L-systèmes et les cellworks L-systèmes.

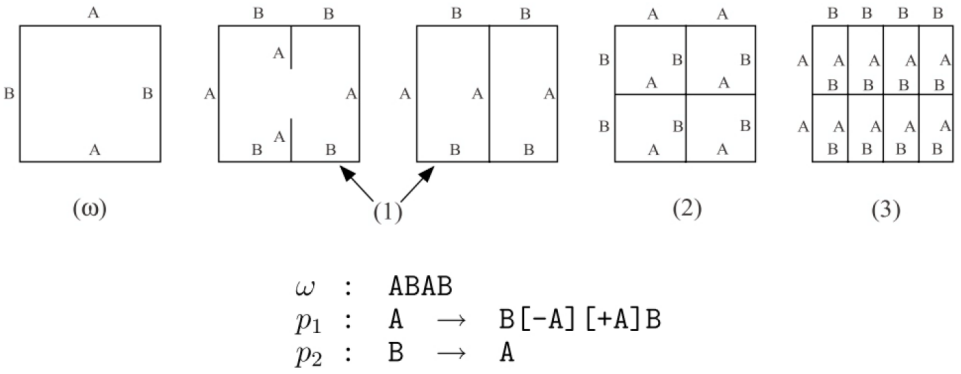


FIGURE 1.32 – Exemple de map L-système [PL96]. Au cours de la première dérivation, une distinction se fait entre la phase de réécriture d'arêtes et la connexion d'arêtes pendantes.

Les maps L-systèmes permettent d'étendre le concept des L-systèmes à la dimension 2 (voir figure 1.32, et ont été utilisés dans [PL96] pour simuler les divisions cellulaires. Les cellworks L-systèmes (ou 3D maps L-systèmes) sont quant à eux une extension du concept des L-systèmes adaptée aux cellules de dimension trois. Dans cette extension, les "cellworks" sont des ensembles de cellules, et chaque cellule est assimilée à un volume. La figure 1.33 illustre un exemple de résultat obtenu par l'application d'un cellwork L-système pour simuler le développement d'une cellule épidermique.

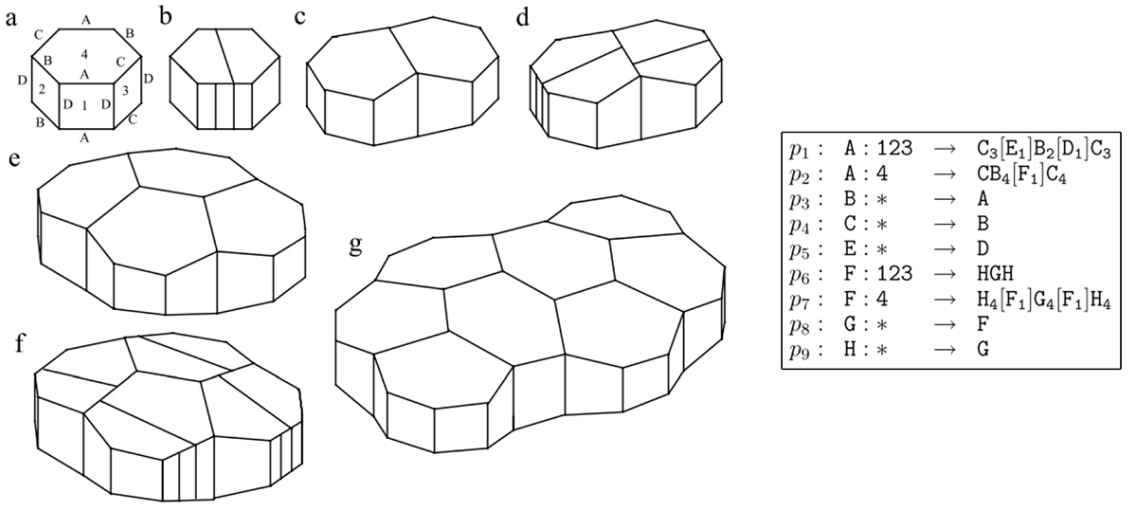


FIGURE 1.33 – Exemple de cellwork L-système [LR78] représentant le développement d'une cellule épidermique : (a) l'ensemble de cellules initial ; (b), (d) et (f) les ensembles de cellules obtenu après une division cellulaire ; et (c), (e) et (g) les ensembles de cellules arrivé à l'état d'équilibre après division.

Les maps L-systèmes et les cellworks L-systèmes se sont révélés utiles dans de nombreuses applications. Cependant, ces deux extensions présentent une lacune significative. En effet, la spécification des grammaires est difficile, car toutes les opérations sont définies sur les arêtes, et qu'aucune interaction entre les cellules n'est supportée. C'est pour pallier cette limitation que les 3G-cartes L-systèmes ont été introduits.

1.4.3 3G-cartes L-systèmes

Le concept de 3G-carte L-système (ou 3Gmap L-system) est une une extension des L-systèmes qui est basée sur l'utilisation des cartes généralisées de dimension trois, et qui a été introduite par Terraz *et al.* dans [TGM+09] pour apporter une solution aux limitations des maps L-systèmes et des cellworks L-systèmes. Les travaux présentés dans ce mémoire sont basés sur ce modèle de représentation qui, contrairement à ce que son nom l'indique, n'utilise pas des grammaires de L-systèmes classiques, mais des grammaires de Chomsky [Cho57]. Ainsi, contrairement aux autres types de L-systèmes présentés jusqu'à présent, plutôt que d'appliquer les règles de production en parallèle à chaque itération, les 3G-cartes L-systèmes les appliquent de manière séquentielle.

Ce modèle de représentation est intuitif, car il permet de contrôler la forme et la structure interne d'objets 3D à l'aide d'opérations haut-niveau agissant directement sur les volumes et non sur les arêtes comme dans les maps et les cellworks L-systèmes. Ces opérations haut-niveau sont capables de coller deux volumes entre eux, de scinder un volume en deux ou d'ajouter un nouveau volume sur un volume existant. De plus, la notion de voisinage entre volumes est introduite notamment grâce aux labels, ce qui permet par exemple de gérer le comportement de la croissance d'une structure en fonction du contexte. Pour mieux appréhender ce concept, nous allons maintenant définir les notions importantes qui sont nécessaires à la compréhension de ce mémoire.

1.4.3.1 Notations

Labels de volumes. L'objectif est de simplifier le processus de modélisation en définissant des règles de production qui agissent directement sur les volumes. Pour ce faire, à chaque volume est assigné un nom (ou label) permettant de les identifier. Dans une grammaire, un label de volume s'écrit en lettres capitales. Ainsi, le symbole A représente un volume de label A.

Types de volumes. Pour faciliter le contrôle du processus de modélisation, les 3G-cartes L-systèmes utilisent des prismes comme volumes. Grâce à ces derniers, il est ainsi facile de garder une information sur l'orientation du volume. En effet, la base du prisme représente alors l'extrémité inférieure, alors que la face qui se trouve à l'opposé de la base est l'extrémité supérieure. L'ordre n d'un prisme est défini par l'ordre de sa base ($n = 4$ pour un cube par exemple). Pour représenter un prisme de label A et d'ordre n, on utilise la notation $A(n)$.

Labels de faces. Toujours dans l'optique de simplifier le contrôle du processus de modélisation, un label est assigné à chaque face d'un volume (voir figure 1.34). Ainsi, pour un volume d'ordre n, on a les labels suivants :

- O désigne la base du prisme, appelée face origine ;
- E désigne l'extrémité supérieure du prisme, appelée face extrémité ;
- $C1, C2, \ldots, Cn$ sont les faces latérales du volume. L'ordre de labellisation de ces faces se fait dans un ordre trigonométrique par rapport à la normale de O.

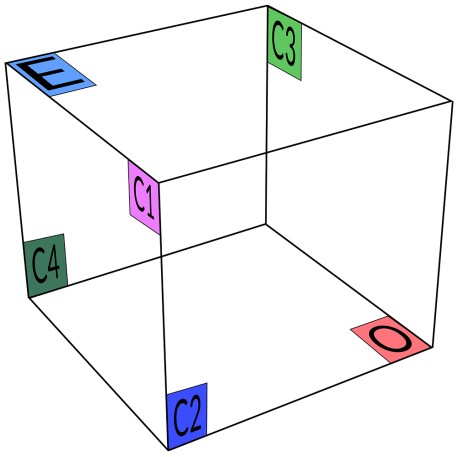


FIGURE 1.34 – Illustration de la labellisation des faces d'un prisme d'ordre 4.

Sélection de faces. Au sein d'une règle de production, il est possible de sélectionner une face. Pour sélectionner la face E d'un volume A par exemple, il suffit d'écrire A_E. En plus de cette notation, l'utilisateur peut sélectionner un groupe de faces à l'aide du symbole $*$. Ainsi, pour un prisme A on a :

- A_* qui représente toutes les faces de A ;
- A_{C*} qui sélectionne toutes ses faces latérales ;
- A_{C*-2} qui représente toutes les faces latérales, sauf la face $C2$.

Relation d'adjacence entre volumes. Deux faces A_x et B_y sont adjacentes si elles ont une arête en commun. De plus, deux volumes A et B sont adjacents si ils ont au moins une face adjacente (voir figure 1.35). Cette relation d'adjacence peut être représentée par les symboles $<$ et $>$, qui sont également utilisés dans les L-systèmes dépendants du contexte. N'importe lequel de ces symboles peut être utilisé pour représenter l'adjacence, car celle-ci est symétrique. Ainsi :

- $A_E > B_O$ et $A_E < B_O$ représentent la relation d'adjacence entre la face extrémité de A et la face origine de B ;
- $A_* > B_{C2}$ et $A_* < B_{C2}$ représentent la relation d'adjacence entre n'importe quelle face de A avec la face $C2$ de B ;
- $A_* > B_*$ et $A_* < B_*$ permettent d'exprimer la relation d'adjacence entre A et B par n'importe laquelle de leurs faces.

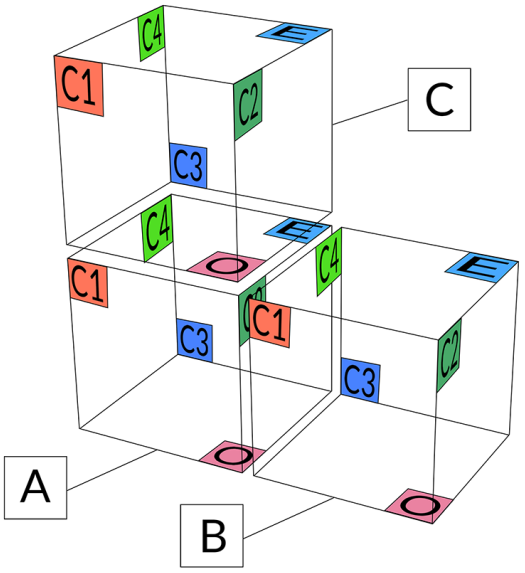


FIGURE 1.35 – Relation d'adjacence : ici, C est adjacent à B par les faces $C2$ et E.

Syntaxe des règles et dépendance du contexte. Grâce à la notion d'adjacence entre volumes, il est possible de faire croître un système en prenant en compte son contexte. Pour représenter la dépendance du contexte, les règles de production respectent la syntaxe suivante (tiré du formalisme *CPFG* [PHM99]) :

$$predecesseur : \ bloc_1 \ cond \ bloc_2 \ \rightarrow \ successeur, \text{ où}$$

- $predecesseur$ est le volume sur lequel la règle $successeur$ sera appliquée ;
- $cond$ est une condition qui définie si la règle doit être appliquée ou pas ;
- $bloc_1$ est optionnel, et représente un ensemble de lignes de code qui sont toujours appliquées ;
- $bloc_2$ est optionnel, et représente un ensemble de lignes de code qui seront appliquées seulement si la condition $cond$ est vérifiée.

Par exemple, la règle $A : A_O < D_E \ and \ A_E > E_O \rightarrow BC$ signifie que la règle de production $A \rightarrow BC$ sera appliquée seulement si A_O et D_E sont adjacentes et si A_E et E_O sont adjacentes.

1.4.3.2 Interprétation géométrique

Comme nous avons vu dans la section 1.4.2.2, pour comprendre comment passer d'une grammaire ne contenant que du texte à un modèle 3D, il nous faut un moyen d'interpréter graphiquement le résultat des dérivations du système. Dans le cas des 3G-cartes L-systèmes, cette opération s'appelle le plongement.

Pour pouvoir calculer le plongement d'un 3G-carte L-système, chaque volume est défini par un ensemble d'attributs (ou paramètres). Ces attributs sont : l'ordre du prisme (O), les angles de rotation de la tortue LOGO (atx, aty, atz), les dimensions du volume (H, L, W)

et un facteur de translation (ct). Avec l'ajout de ces paramètres, la définition d'un volume A par exemple devient alors : $A(O, ct, atx, aty, atz, H, L, W)$.

À chaque dérivation du système, le plongement de chaque volume est calculé en prenant en compte tous ses attributs géométriques. Ainsi, l'algorithme permettant de calculer la géométrie d'un volume A par exemple se déroule de la manière suivante :

1. Calculer la position et l'orientation de A dans l'espace en utilisant ses angles de rotation (atx, aty, atz) ainsi que son facteur de translation (atx).
2. Calculer la forme par défaut de A en utilisant ses attributs de dimension (H, L, W), et sans prendre en compte son voisinage. Par exemple, un volume $A(4, 0, 0, 0, N, N, N)$ donnera un cube de côté N.
3. Prendre en compte le voisinage pour déterminer la forme finale de A. La position finale de chaque sommet est définie en calculant le barycentre de tous les sommets correspondants associés à ses volumes incidents.

Sur la figure 1.36, nous pouvons voir ce qui se passe lorsqu'on calcule le plongement du modèle avant, et après qu'un volume B ait été défini comme adjacent à un volume A. Au début, A garde sa forme par défaut, soit un cube dans notre exemple. Et une fois que B est défini comme étant adjacent à A, on calcule le barycentre des sommets incidents aux deux faces collées.

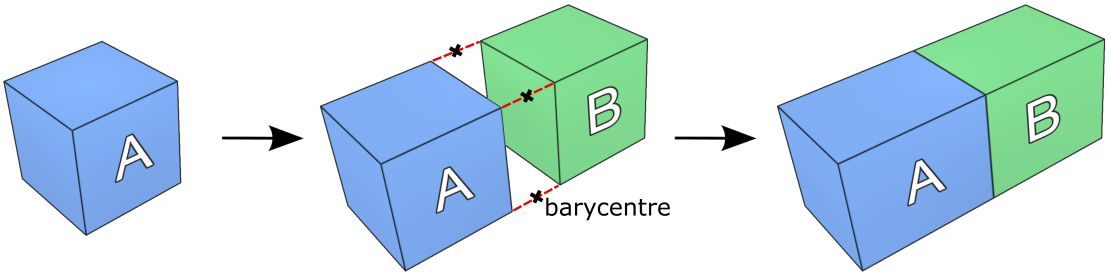


FIGURE 1.36 – Interprétation géométrique. Illustration du calcul du plongement lorsqu'un volume B est adjacent à un volume A.

1.4.3.3 Règles de base et opérations topologiques correspondantes

Comme expliqué au début de cette section, les 3G-cartes L-systèmes utilisent des opérations haut-niveau qui agissent directement sur les volumes. Nous pouvons trouver des opérations qui font croître le modèle en y ajoutant des volumes, d'autres qui collent des volumes entre eux le long d'une de leurs faces ou bien encore qui divisent un volume en deux volumes de labels différents. Tout comme les L-systèmes classiques, ces opérations sont définies dans des règles de production que nous allons décrire ci-après.

Scission de volumes. La première règle de base est une règle qui permet d'éclater un volume en deux volumes de labels différents. Le prototype de la règle de scission de volumes est défini de cette façon :

$$A \rightarrow^{O} BC \tag{1.1}$$

Dans cet exemple, on crée deux volumes B et C à partir d'un volume A, en prenant A_O comme face de support. Pour ce faire, on coupe dans un premier temps toutes les faces adjacentes à A_O, puis on ferme les deux volumes B et C ainsi créés à l'aide de faces ayant le même degré que A_O. La figure 1.37 illustre le résultat de l'application de la règle 1.1.

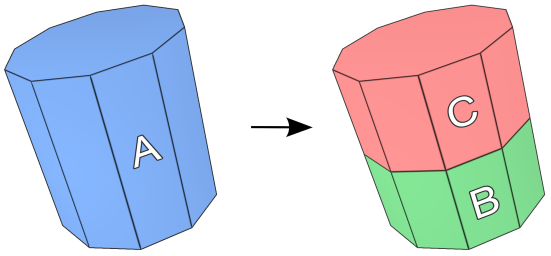


FIGURE 1.37 – Règle de scission. À droite, le résultat de l'application de la règle 1.1 appliquée au volume de gauche.

Croissance. Le second type de règle utilisé est la croissance, appelée aussi règle d'ajout de volume. Comme son nom l'indique, cette règle va permettre de faire croître le modèle en créant un volume qu'on va ajouter sur la face support d'un autre volume, sous réserve que cette dernière n'est pas déjà collée à une autre face. Une règle d'ajout est définie comme suit :

$$A \rightarrow A[B(n)]_E \tag{1.2}$$

Ici, cette règle permet de créer un volume B d'ordre n qui va être collé sur la face E du volume A (figure 1.38). Le nouveau volume est collé à la face support A_E le long de sa face origine, soit B_O dans notre exemple. Si les deux faces à joindre n'ont pas le même ordre, alors on fait fendre les arêtes de la face avec le plus petit ordre jusqu'à obtenir le même nombre d'arêtes que celle avec le plus grand ordre. Par contre, si le degré de B n'est pas spécifié dans la règle, alors B est un prisme d'ordre égal au degré de A_E.

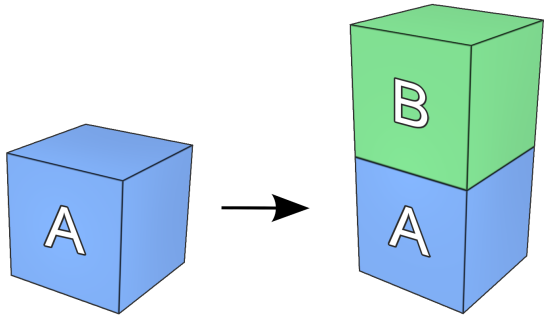


FIGURE 1.38 – Règle d'ajout. Illustration du résultat de l'application de la règle 1.2 sur un volume A.

Collage. Le dernier type de règle est le collage. Les règles de collage permettent de coller deux volumes le long d'une de leurs faces sous réserve que ces faces soient au moins adjacentes par une arête et libres. Voici le prototype d'une règle de collage :

$$B : B_{C2} < C_{C4} \rightarrow B_{C2}|C_{C4} \tag{1.3}$$

Dans cet exemple, on vérifie dans un premier temps si la face $C2$ du volume B est adjacente à la face $C4$ du volume C. Si c'est le cas, alors on colle ces deux faces. La figure 1.39 illustre le résultat de l'application de cet exemple.

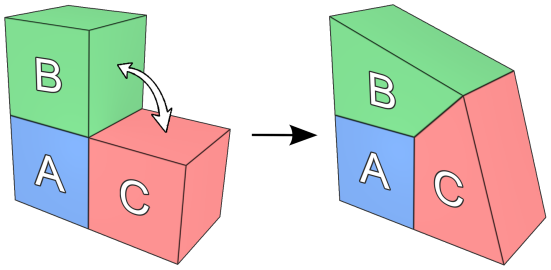


FIGURE 1.39 – Règle de collage. Application de la règle 1.3 permettant de coller les volumes B et C entre eux.

1.4.3.4 Fonctionnement d'une grammaire

Pour mieux comprendre la structure d'une grammaire de 3G-cartes L-systèmes, nous allons prendre un exemple simple tiré de [TGM+09], qui génère la spirale illustrée sur la figure 1.40.

Grammaire 1.1 : `Grammaire tirée de [TGM+09]`

@définition des volumes@
$\#define\ A(16,1,0,0,0,5,5,5)$
$\#define\ B(4,1,0,0,0,0.5,0.4,1)$

@définition des variables@
$\#define\ v = 0.5$

@définition de l'axiome@
$\#axiome :\ A$

@Règles de production@
$p01\ A\{v = v + 5\}\{etape \geq 0\}\{\} \rightarrow A[A(,,,20,,<5+v>,<5+v>,<5+v>)]_E$
$p02\ A\{\}\{etape \geq 0\}\{\} \rightarrow A[B(,,,,,<(etape/3)+v>,,)]_{C2,C4,C6,C8,C10,C12,C14,C16}$
$p03\ B\{\}\{etape \geq 0\}\{\} \rightarrow B_{C*}|B_{C*}$

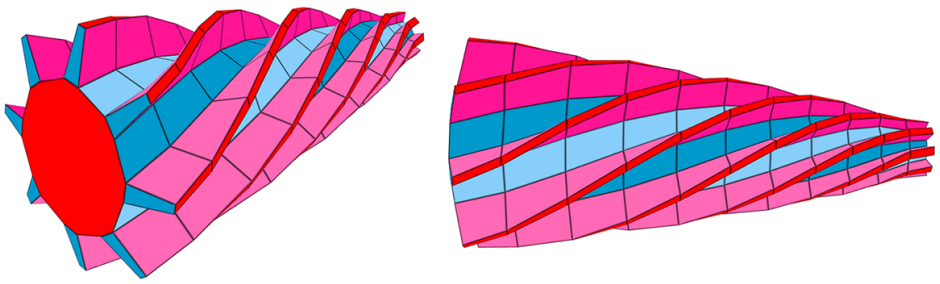


FIGURE 1.40 – Illustration d'une spirale, résultat obtenu après 10 dérivations de la Grammaire 1.1.

À l'instar d'un L-système "classique", nous pouvons voir qu'une grammaire est composée de quatre parties distinctes, délimitées dans notre exemple par les commentaires compris entre les symboles @.

Définition des volumes. Premièrement, nous devons définir les volumes qui vont intervenir dans la modélisation de l'objet. Pour faire l'analogie avec les L-systèmes classiques, nous pouvons dire qu'il s'agit ici de la définition de l'alphabet du système. Ainsi, pour définir un volume A avec ses paramètres, on écrit : $\#define\ A(O, ct, atx, aty, atz, H, L, W)$. Comme expliqué dans la section 1.4.3.1, le nom d'un volume est toujours spécifié en lettres capitales.

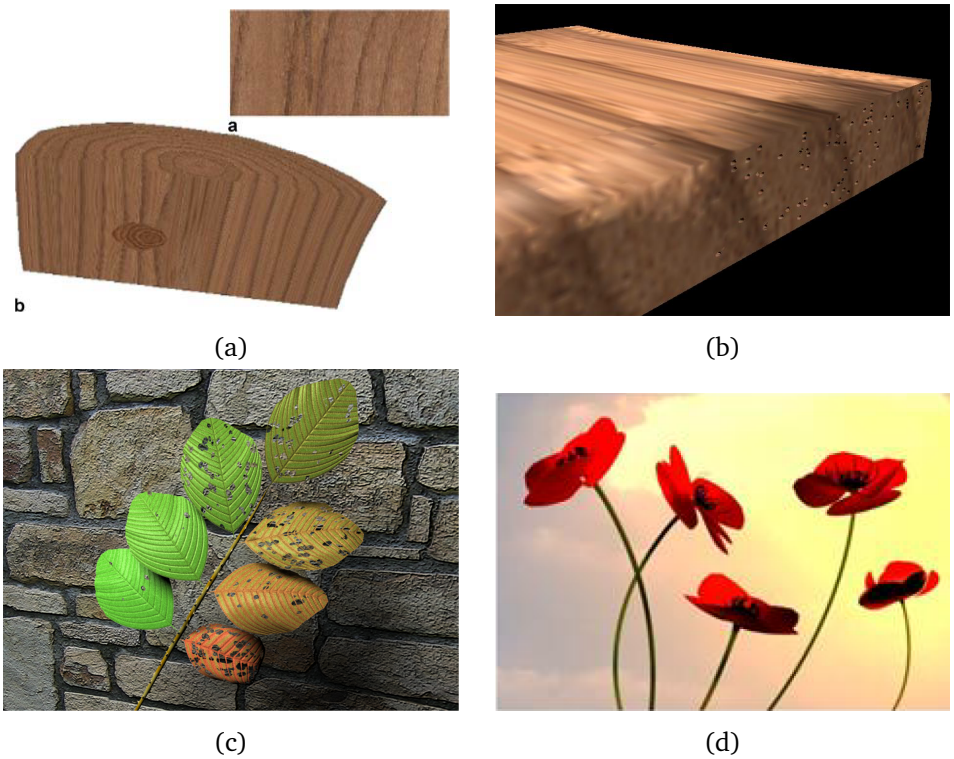
Définition des variables. Une fois que les volumes ont été définis, les variables sont spécifiées. Ces dernières sont capables d'influencer le processus de modélisation. Pour ce faire, une variable est dans un premier temps initialisée grâce à l'instruction $\#define$. Le nom d'une variable étant toujours spécifié en lettres minuscules, la définition d'une variable m de valeur égale à 5 s'écrit : $\#define\ m = 5$. Par la suite, au sein des règles de production, cette variable pourra être utilisée dans des blocs particuliers appelés blocs de script. Ces blocs, délimités par "<" et ">", incluent du code Lua, un langage de script qui nous permet d'utiliser des fonctions mathématiques et de combiner nos variables.

Définition de l'axiome. Lorsque les variables ont été spécifiées, on définit l'axiome de départ choisi parmi l'ensemble des volumes définis plus haut. Ce volume représentera l'état initial du système. Dans la grammaire 1.1, l'axiome de départ est le volume A.

Définition des règles. Enfin, une fois que tous les paramètres du système ont été définis, on décrit toutes les règles de production. Comme expliqué au début de cette section, la partie gauche d'une règle est composée des blocs conditionnels $bloc1$, $cond$ et $bloc2$. Ici, chacun de ces blocs sont délimités par "{" et "}". De plus, nous pouvons voir dans les règles $p01$ et $p02$ que les attributs des volumes à ajouter sont redéfinis. Il s'agit ici du mécanisme de surcharge qui permet, lorsqu'un volume est créé, de lui donner des valeurs différentes de celles qui ont été définies au début de la grammaire. Cette nouvelle valeur peut être égale à une valeur constante, ou au résultat renvoyé par du code contenu dans un bloc de script. Dans la grammaire 1.1, la règle $p01$ est une règle d'ajout. À chaque dérivation, elle permet d'incrémenter la valeur de v et d'ajouter un volume A sur la face E de tous les volumes A. Chaque nouveau volume A est plus gros que son volume support (ou volume père) et effectue une rotation de 20 degrés autour de l'axe $\vec{H}$ de la tortue LOGO avant d'être collé à son père. Dans les grammaires, une variable globale appelée $etape$ représente le nombre de dérivations subies par le système depuis sont état initial. Ainsi à chaque dérivation, la règle $p02$ va ajouter un volume B ayant une hauteur $H = (etape/3) + v$ sur une partie des faces latérales de tous les volumes A. Enfin, la règle $p03$ est en charge du collage de tous les volumes B ainsi créés, le long de leurs faces côtés si celles-ci sont adjacentes.

1.4.3.5 Quelques extensions...

Le concept des G-cartes L-systèmes s'est révélé très pertinent dans la représentation d'objets naturels. En effet, il fut introduit dans un premier temps dans [TGM+09] pour modéliser la structure interne du bois. Grâce à l'aspect volumique du concept, les auteurs ont réussi à réaliser du rendu basé sur des volumes élémentaires, permettant ainsi d'obtenir une apparence réaliste lors de la découpe d'un modèle 3D de bois (voir figure 1.41a). La modélisation de bois par les 3G-cartes L-systèmes a ensuite été étendue par Lam *et al.* dans [LTBG09], permettant ainsi de simuler le vieillissement du bois dû à l'attaque d'insectes ou aux craquelures. Ainsi, en assignant cette fois-ci des attributs aux faces, comme l'orientation de celles-ci par rapport au modèle entier par exemple, les auteurs ont pu représenter des morceaux de bois attaqués par différents insectes (voir figure 1.41b). Peyrat *et al.* quant à eux introduisent les 2G-cartes L-systèmes paramétriques dans [PTMG08] pour la génération de grandes variétés de feuilles. Grâce aux variables, aux blocs de scripts et à la surcharge, les auteurs ont réussi à générer des atlas de feuilles de formes et d'âges différents à l'aide d'une seule grammaire (voir figure 1.41c). Les 3G-cartes L-systèmes ont également été utilisées par Petrenko *et al.* dans [PTSG11] pour modéliser des fleurs de manière interactive (voir figure 1.41d). Les auteurs présentent des nouvelles extensions comme : la possibilité d'assigner un matériau à chaque volume, le fait de "casser" une grammaire volumineuse en plusieurs petites grammaires (ou modules), ainsi que le mécanisme d'interactivité permettant de rendre le processus d'écriture de grammaires plus intuitif en modifiant interactivement la valeur des variables du système.



(a) (b)

(c) (d)

FIGURE 1.41 – Illustration des résultats obtenus par (a) Terraz *et al.* dans [TGM+09] , (b) Lam *et al.* dans [LTBG09], (c) Peyrat *et al.* dans [PTMG08], et (d) Petrenko *et al.* dans [PTSG11].

Chapitre 2

Génération de fruits et de leur structure interne

Sommaire

Chapitre 2

Génération de fruits et de leur structure interne

2.1 Introduction

En nous basant sur l'étude des travaux antérieurs, nous pouvons constater que la représentation de fruits est un domaine qui a été encore relativement peu abordé en informatique graphique. Néanmoins, des résultats convaincants ont été réalisés, permettant ainsi de répondre en partie, à certaines problématiques liées par exemple à la simulation du vieillissement d'un fruit [KRB11, LCW+12] ou au rendu de sa structure interne [TSNI10, TOII08]. Nous pouvons également constater à partir de cet état de l'art que les méthodes procédurales, et plus particulièrement les L-systèmes, sont très adaptées à la représentation d'éléments naturels. Par exemple, c'est en utilisant des open L-systèmes que les travaux de Huang *et al.* [HJT+13] ont permis de générer des grappes de raisin de manière réaliste. Cependant, parmi les travaux identifiés dans le chapitre précédent, aucun n'est axé spécifiquement sur la génération de fruits et de leur structure interne. Le fruit étant géré par une énorme quantité de processus complexes, il semble en effet difficile d'appréhender facilement sa modélisation.

Dans cette partie de nos travaux, nous avions plusieurs objectifs principaux. Tout d'abord, la facilité de conception d'un fruit au niveau de sa forme. Ensuite, l'identification des éléments clés de sa structure interne pour pouvoir représenter cette dernière de manière relativement simple et intuitive. Enfin, la capacité de générer une grande variété de formes de fruits différentes pour une espèce donnée. Pour ce faire, nous avons développé un modèle reposant sur les 3G-cartes L-systèmes paramétriques. Cet outil, introduit par Terraz *et al.* [TGM+09], est la fusion des L-systèmes [Lin68] et des 3G-cartes [Lie91], et nous permet de facilement définir des grammaires qui décrivent la topologie de structures volumiques complexes. De plus, en y ajoutant un certain nombre d'extensions, ce modèle nous a permis de générer des fruits ainsi que leur structure interne de manière relativement simple.

Ces travaux ont été présentés lors des 27èmes journées de l'AFIG et lors de la conférence *Computer Graphics International 2015*, et publiés dans la revue internationale *The Visual Computer* [BTG15].

2.2 Problématique

Au premier abord, la génération automatique de fruits semble être une tâche difficile à accomplir, notamment à cause du très grand nombre d'espèces que nous pouvons trouver dans la nature. De plus, cette diversité n'en devient que plus accentuée si nous nous plaçons à l'échelle de l'espèce. En effet, si nous prenons deux pommes par exemple, ces fruits vont être similaires mais pas totalement identiques. Ainsi, malgré des caractéristiques spécifiques à leur espèce (exemple : la couleur rouge), chacun de ces fruits présentera des variations qui lui sont propres (exemple : un rouge plus intense chez la première).

Cependant, malgré ces différences visibles, une grande majorité des fruits est composée d'une structure interne relativement similaire. En effet, 90% des fruits présents dans la nature sont des fruits simples charnus, alors qu'une grande partie des 10% restants consistent en un agglomérat de fruits de ce type. Cette particularité fait qu'en général, la structure interne d'un fruit sera composée exclusivement d'un carpelle (ovaire), ou de plusieurs carpelles soudés autour d'un axe, ce qui facilite grandement l'approche de sa modélisation. Ainsi, afin de couvrir le plus grand nombre d'espèces possible, nous avons donc décidé de focaliser nos travaux sur la génération de fruits simples charnus.

La majorité des fruits présente donc une symétrie rotationnelle par rapport à un axe, autour duquel sont soudés les carpelles. Ainsi pour générer un fruit 3D, nous voulons dans un premier temps que sa forme générale suive l'orientation de cet axe. Ensuite, une fois que ce dernier sera créé, les carpelles viendront s'y greffer pour former le volume final. Pour représenter cette structure arborescente, nous avons utilisé un 3G-carte L-système. Cet outil nous offre la possibilité de construire n'importe quel agencement de volumes dans l'espace 3D et ce, de manière relativement aisée.

Une autre facette que nous voulions aborder est la variation de la forme finale du fruit. En effet, à partir du moment où nous avons réussi à obtenir la forme générale d'une espèce de fruit, nous voulons être capables de représenter les variations géométriques qu'il est possible d'observer parmi les fruits de cette espèce. Pour répondre à cette problématique, nous avons paramétré notre 3G-carte L-système de manière à générer sa forme à l'aide de fonctions mathématiques. Ces dernières nous offrent ainsi la possibilité de faire varier la forme générale en utilisant des paramètres simples.

Enfin, le dernier aspect que nous avons abordé est la représentation de la structure interne. Lorsque le pistil de la fleur se transforme en fruit, les parois des ovaires (carpelles) maturent pour former les trois couches du péricarpe (exocarpe, mésocarpe et endocarpe). En plus de ces trois couches, nous voulons que le modèle puisse représenter un composant important de la structure interne : les graines. Pour ce faire, nous avons introduit une extension aux 3G-cartes L-systèmes qui est basée sur la subdivision volumique et la subdivision surfacique. Grâce à cette amélioration, la forme d'un fruit peut alors être subdivisée pour créer ses graines ainsi que les différentes couches de sa structure interne.

La figure 2.1 permet de mieux visualiser le pipeline de notre méthode. L'utilisateur fournit une grammaire accompagnée de différents paramètres tels que le nombre de carpelles, la taille du fruit ou le nombre de graines par exemple. Notre modèle crée alors dans un premier temps la forme standard du fruit. Ensuite, il génère des variations sur cette forme. Enfin, les éléments de la structure interne sont créés, composant ainsi la structure

volumique finale. En fonction des besoins de l'utilisateur, le modèle de fruit complet résultant (surface et structure interne) pourra être utilisé par la suite pour faire des mesures, de la simulation de vieillissement, etc. . .

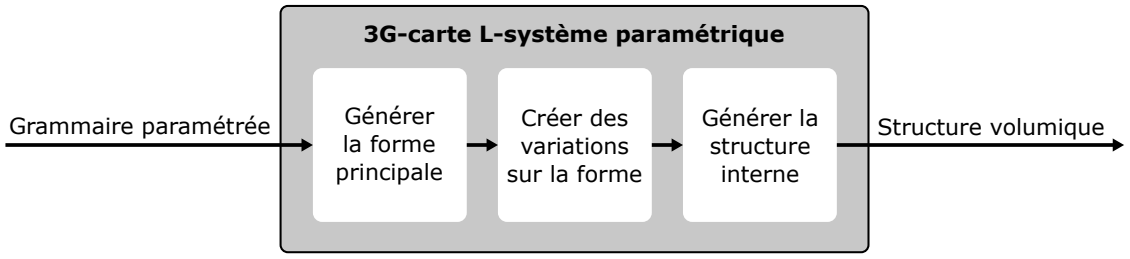


FIGURE 2.1 – Les étapes de l'exécution de notre modèle de génération de fruits.

2.3 Modélisation de la surface d'un fruit

Dans cette section, nous présenterons les deux premières étapes de notre méthode, à savoir : la génération d'une forme de fruit générique et l'application de variations à cette forme. Ainsi, nous verrons dans un premier temps comment utiliser un 3G-carte L-système pour modéliser un fruit en respectant sa structure macroscopique. Ensuite, nous allons introduire une paramétrisation à notre modèle dans le but de pouvoir définir la forme du fruit à l'aide de courbes mathématiques. Dans la dernière partie de cette section, nous verrons enfin comment faire varier la forme de ces courbes dans le but d'appliquer des variations globales sur ce fruit générique et créer ainsi des espèces de fruits de formes variées.

2.3.1 Création d'un 3G-carte L-système générique de fruit

La majorité des fruits présente une structure similaire, basée sur une symétrie rotationnelle atour d'un axe. Afin de refléter au mieux la réalité botanique, nous avons décidé de nous baser sur cette propriété pour créer un 3G-carte L-système permettant de modéliser la forme d'un fruit. Ce concept se prêtant très bien à la création de structures arborescentes, les étapes de l'exécution de notre système présentent quelques similarités avec les méthodes de modélisation d'arbres [Pru04]. En effet, la génération d'un fruit à l'aide de notre modèle s'articule autour de la croissance d'un axe central, similaire à un tronc donc, sur lequel vont croître des axes de premier ordre, qui peuvent être vus comme des branches (figure 2.2). La différence cependant avec les modèles de génération d'arbres réside dans le fait que tous les axes de premier ordre sont ensuite collés entre eux dans le but de fermer la structure volumique et former ainsi le péricarpe du fruit.

Nous allons voir à présent une première version de la grammaire de notre 3G-carte L-système, qui permet de générer le modèle de tomate illustré sur la figure 2.2. Pour une meilleure compréhension des étapes de l'exécution du système, les résultats des dérivations y sont illustrés en 2D.

Grammaire 2.1 : `Grammaire simple générant les résultats de la figure 2.2`

@définition des volumes@
$\#define\ A(10,1,0,0,0,1,3,3)$
$\#define\ B(4,1,0,0,0,30,5,5)$

@axiome@
$\#axiome :\ A$

@règles de production@
$p01\ A\{\}\{etape == 1\}\{\} \rightarrow A[A(,,,,,5,,)]_E$
$p02\ A\{\}\{etape == 2\}\{\} \rightarrow A[A(,,,,,10,,)]_E$
$p03\ A\{\}\{etape == 3\}\{\} \rightarrow A[A(,,,,,15,,)]_E$
$p04\ A\{\}\{etape == 4\}\{\} \rightarrow A[A(,,,,,20,,)]_E$
$p05\ A\{\}\{etape == 5\}\{\} \rightarrow A[A(,,,,,15,,)]_E$
$p06\ A\{\}\{etape == 6\}\{\} \rightarrow A[A(,,,,,5,,)]_E$
$p07\ A\{\}\{etape == 7\}\{\} \rightarrow A[A(,,,,,5,,)]_E$
$p08\ A\{\}\{etape == 8\}\{\} \rightarrow A[A(,,,,,1,,)]_E$

$p09\ A\{\}\{etape == 0\}\{\} \rightarrow A[B(,,,,,1,,1)]_{C*}$
$p10\ A\{\}\{etape == 1\}\{\} \rightarrow A[B(,,,,,30,,5)]_{C*}$
$p11\ A\{\}\{etape == 2\}\{\} \rightarrow A[B(,,,,,40,,10)]_{C*}$
$p12\ A\{\}\{etape == 3\}\{\} \rightarrow A[B(,,,,,50,,15)]_{C*}$
$p13\ A\{\}\{etape == 4\}\{\} \rightarrow A[B(,,,,,55,,20)]_{C*}$
$p14\ A\{\}\{etape == 5\}\{\} \rightarrow A[B(,,,,,50,,15)]_{C*}$
$p15\ A\{\}\{etape == 6\}\{\} \rightarrow A[B(,,,,-5,45,,5)]_{C*}$
$p16\ A\{\}\{etape == 7\}\{\} \rightarrow A[B(,,,,-17,30,,5)]_{C*}$
$p17\ A\{\}\{etape == 8\}\{\} \rightarrow A[B(,,,,-25,5,,1)]_{C*}$

$p18\ B\{\}\{etape == 9\}\{\} \rightarrow B_{C*}|B_{C*}$

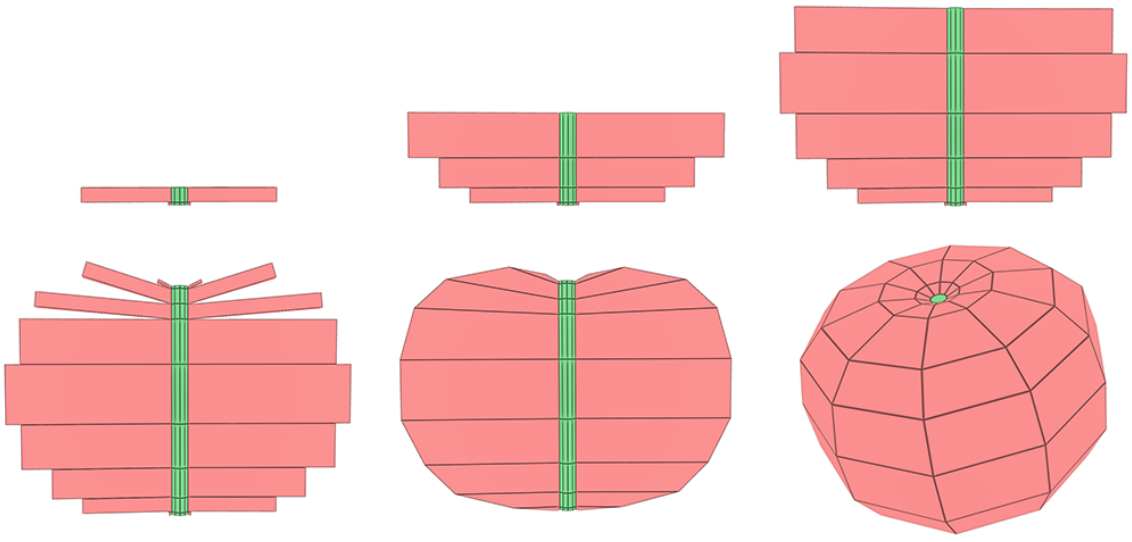


FIGURE 2.2 – Illustration 2D des résultats obtenus après 2, 4, 6, 9 et 10 dérivations de la grammaire 2.1.

Dans le système représenté par la grammaire 2.1, l'axiome de départ est défini par un volume A, sur lequel d'autres volumes de même label vont être ajoutés dans le but de former l'axe central (ou tronc) de l'arborescence. Sur la figure 2.2, ces volumes sont illustrés en vert. Les volumes B quant à eux y sont illustrés en rouge, et représentent les axes de premiers ordre (ou branches) de la structure arborescente. À la fin de l'exécution du système, ces derniers formeront le péricarpe.

Lorsque les volumes et l'axiome de départ sont définis, les règles de production se chargent de générer la forme finale du fruit, tout en respectant le processus de modélisation décrit au début de cette section. Ainsi, à chaque itération, l'une des règles $p01 - p08$ est choisie en fonction du nombre de dérivations déjà subies par le système, pour créer un volume A sur l'extrémité supérieure du tronc généré par les volumes de même label. De plus, toujours en fonction du nombre de dérivations ayant eu lieu, une règle est choisie parmi les règles $p09 - p17$ pour créer les branches de la structure en faisant croître un volume B sur chaque face latérale du volume A fraîchement créé. Enfin, une fois que la structure finale du fruit a été correctement créée, la règle $p18$ se charge de dessiner la forme finale du fruit en collant entre eux tous les volumes B qui sont adjacents. La partie inférieure de la figure 2.2 illustre cette étape de collage.

La force de ce système réside dans le fait que la forme générale du fruit n'est influencée que par deux facteurs principaux : l'axe central, responsable de son allongement et de son orientation, et les axes de premier ordre, qui définissent la courbure de sa surface. Grâce à cette caractéristique, la création d'un fruit devient alors relativement aisée et, en modifiant les attributs des bons volumes, il est possible de lui donner n'importe quelle forme.

Afin de démontrer la généricité de notre méthode, nous nous sommes servis de la grammaire 2.1, qui permet normalement de modéliser une tomate, pour générer deux autres espèces de fruits simples : une poire (figure 2.3) et une banane (figure 2.4). Pour ce faire, nous avons gardé la structure générale de la grammaire 2.1 à laquelle nous avons apporté relativement peu de changements. Dans le cas de la poire par exemple, la forme générale étant plus complexe que celle d'une tomate, nous avons augmenté la précision de la structure arborescente en ajoutant plus de volumes A à l'axe central. La taille et l'orientation des volumes B créés à chaque dérivation est alors adaptée pour que la forme de la surface ressemble à celle d'une poire. Pour ce qui est de la banane, contrairement aux deux autres espèces, celle-ci présente la particularité d'être courbée sur elle-même. Cette propriété est facilement représentable grâce à l'axe central. Pour ce faire, il suffit en effet d'appliquer une rotation aux volumes A qui sont créés à chaque dérivation. Les figures 2.3 et 2.4 permettent de visualiser les résultats obtenus au cours des dérivations du système.

Bien que pertinent pour représenter la forme de différentes espèces, le système représenté par la grammaire 2.1 présente cependant un inconvénient majeur. En effet, les attributs des nouveaux volumes sont définis "en dur" dans la grammaire, et le fait de devoir les modifier pour obtenir une forme différente peut vite devenir compliqué si le fruit à modéliser nécessite une très haute précision, c'est-à-dire beaucoup de volumes. Pour pallier ce problème, nous avons introduit une paramétrisation de la grammaire 2.1 de manière à définir sa forme, non pas à l'aide de constantes, mais à l'aide de fonctions mathématiques.

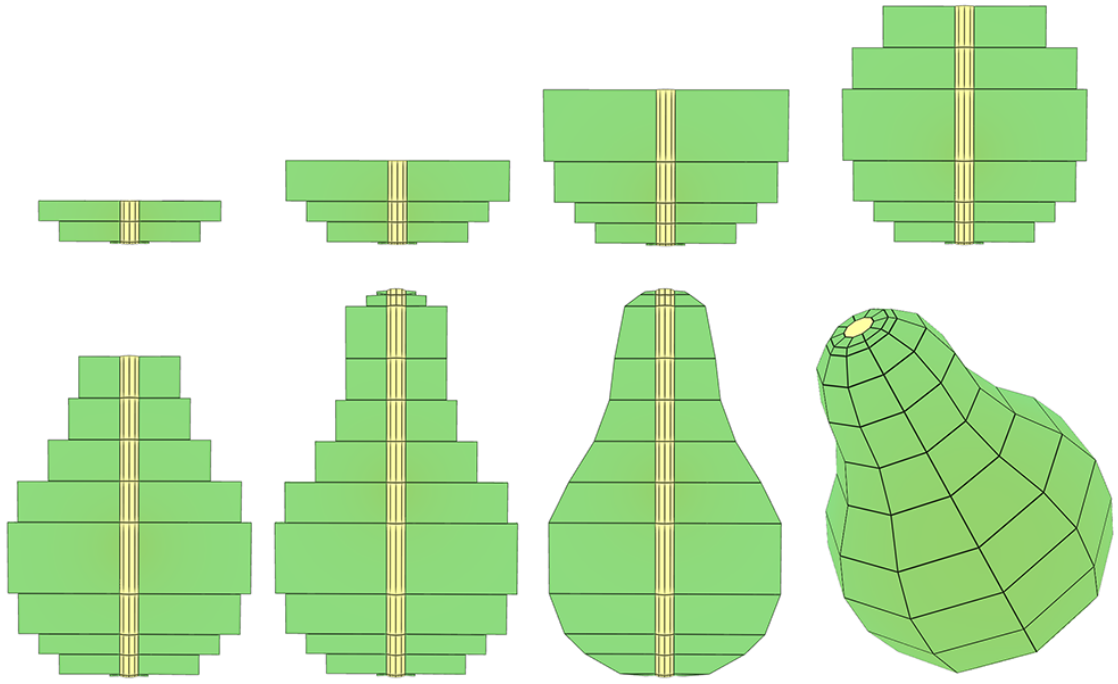


FIGURE 2.3 – Résultats obtenus après 3, 4, 5, 7, 9, 12 et 13 dérivations d'une variante de la grammaire 2.1 pour générer une poire.

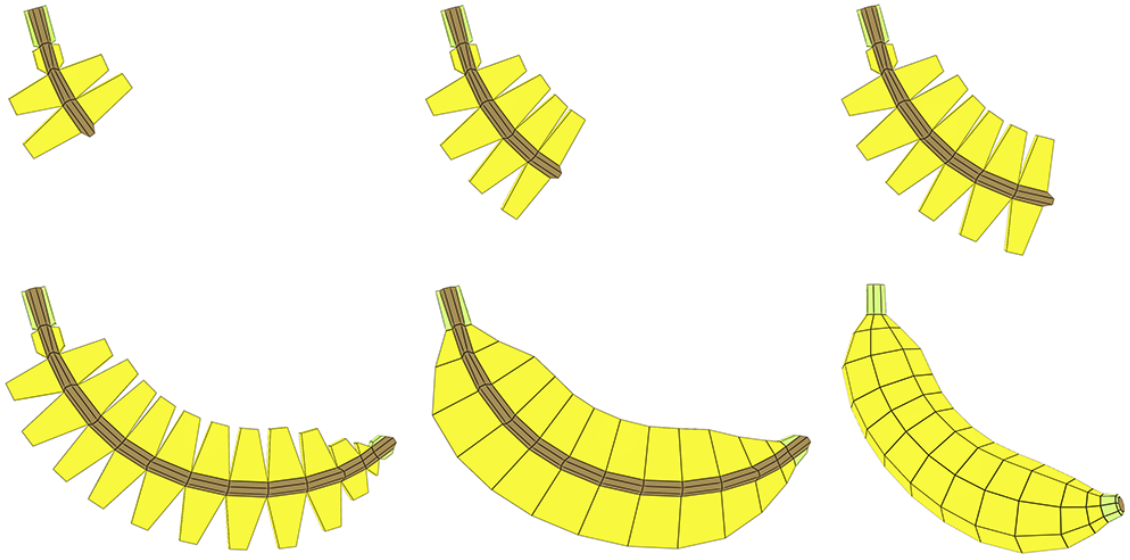


FIGURE 2.4 – Résultats obtenus après 4, 6, 8, 13 et 14 dérivations d'une variante de la grammaire 2.1 pour générer une banane.

2.3.2 Paramétrisation du 3G-carte L-système générique

Nous voulons que la forme du fruit puisse être définie à l'aide de paramètres simples dans le but de faciliter sa modification. Pour ce faire, nous avons appliqué une paramétrisation au 3G-carte L-système représenté par la grammaire 2.1.

2.3.2.1 Présentation du concept

La paramétrisation permet d'utiliser des variables au sein d'un 3G-carte L-système. Ces variables sont initialisées grâce à l'instruction $\#define$ et peuvent être utilisées dans des blocs de scripts, délimités par '<' et '>', dans le but de récupérer le résultat d'opérations arithmétiques. De plus, afin de ne pas surcharger les règles de production et faciliter ainsi leur compréhension, nous avons introduit la possibilité de définir des fonctions en langage C au sein de nos grammaires. Voici un prototype de définition de fonction :

$$\#function \quad carre(x)\ return(x \times x); \tag{2.1}$$

Ici, la fonction 2.1 pourra ensuite être utilisée au sein des règles de production, soit dans des blocs conditionnels ou soit dans des blocs de scripts, dans le but de récupérer la valeur du carré d'une variable.

2.3.2.2 Définition des fonctions

Pour paramétrer notre système, nous devons décomposer la forme finale du fruit en plusieurs fonctions mathématiques. Nous avons déjà vu dans la section 2.3.1 que la forme dépend du "tronc" et des "branches" de l'arborescence générée par notre système. Ainsi, en se basant sur cette observation, nous avons paramétré notre modèle en utilisant des fonctions qui agissent directement sur les attributs du tronc et des branches. Dans la suite de cette section, nous allons ainsi créer petit à petit la version paramétrique de notre système, en présentant chacune des fonctions mathématiques utilisées.

Précision de l'axe central.
La forme de certains fruits est plus complexe que d'autres et nécessite un modèle 3D plus précis pour être représentée correctement. Dans notre système, la précision de la courbe qui dessine la surface est influencée par la fréquence d'échantillonnage. Cette dernière, représentée par le nombre de branches, peut être accentuée en augmentant le nombre de volumes de l'axe central. En effet, nous avons vu précédemment qu'en utilisant plus de volumes A dans la grammaire 2.1, nous avions à notre disposition plus de branches pour dessiner la forme d'une poire de manière efficace. Ainsi, un premier pas vers la paramétrisation de notre système consiste à y introduire une variable $precision$ qui permet de définir le nombre de volumes à utiliser pour générer l'axe central. La grammaire 2.2 montre l'intégration de cette variable dans notre système et la figure 2.5 illustre le résultat de l'application de cette grammaire.

Grammaire 2.2 : `intégration de la variable` $precision$.

@définition des volumes@
$\#define\ A(10, 1, 0, 0, 0, 10, 10, 10)$

@variables@
$\#define\ precision\ =\ 11$

@axiome@
$\#axiome :\ A$

@règles@
$p01\ A\{\}\{etape < precision - 1\}\{\} \rightarrow A[A]_E$

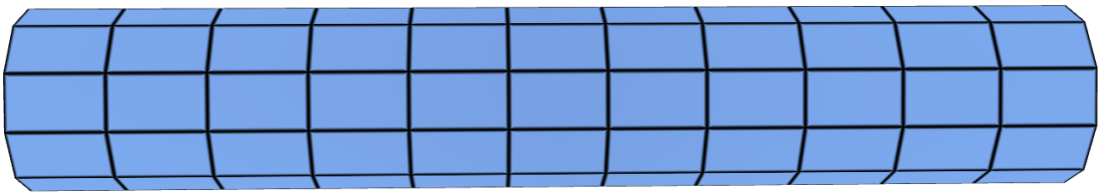


FIGURE 2.5 – Précision de l'axe central. Nous pouvons voir ici le résultat de la grammaire 2.2 pour 10 itérations.

Dans le système représenté par la grammaire 2.2, nous voulons que le nombre de volumes qui forment l'axe central soit égal à la valeur de la variable $precision$. Pour ce faire, la règle $p01$ va créer à chaque itération un nouveau volume A sur l'extrémité du tronc, tant que le nombre de volumes désiré n'est pas atteint. Dans cet exemple, l'axiome de départ faisant partie du tronc, il faudra ainsi effectuer 10 itérations du système pour obtenir un tronc composé de 11 volumes.

Échantillonnage adaptatif du tronc.
Afin d'augmenter le contrôle du résultat final, nous voulons que la courbe qui dessine la forme du fruit soit échantillonnée de manière adaptative. Il y aurait ainsi plus de branches dans les zones où nous avons besoin de détails, et moins de branches dans celles où le détail n'est pas important. Pour ce faire, nous avons utilisé une fonction mathématique qui permet de définir automatiquement la taille des volumes ajoutés au tronc à chaque itération. En effet, comme nous pouvons le constater sur la poire de la figure 2.3, la taille des volumes A influe sur la concentration en branches de chaque zone. Ainsi, les volumes A sont plus grand dans les zones où la forme du fruit n'a pas besoin de détails (exemple : la partie "enflée" de la poire) et plus petits dans celles où la forme doit être précise (exemple : les extrémités de la poire).

La forme d'une tomate présente des exigences similaires à celles d'une poire pour pouvoir être représentée correctement. Ainsi, ses deux extrémités présentant une forte courbure, l'échantillonnage doit être élevé au niveau de ces deux zones. D'autre part, contrairement à ces deux dernières, la partie centrale de la tomate présente de faibles variations

au niveau de sa forme, et pourra ainsi être représentée par un faible nombre de branches. La fonction $f_1(x) = sin(x)$ est bien adaptée pour répondre à ces exigences. En effet, sur l'intervalle $I_1 = [0\,;\pi]$, ses valeurs augmentent dans un premier temps de 0 à 1 sur la première moitié de I_1, et diminuent ensuite de 1 à 0 sur la seconde moitié. La grammaire 2.3 montre comment intégrer cette fonction à notre système, et la figure 2.6 illustre le résultat de l'application de cette grammaire après 10 itérations.

Grammaire 2.3 : `échantillonnage adaptatif du tronc`

@définition des volumes@
$\#define\ A(10, 1, 0, 0, 0, 10, 10, 10)$

@variables générales@
$\#define\ precision\ =\ 11$
$\#define\ longueur\ =\ 15$

@paramétrage de la fonction@
$\#define\ ymin\ =\ 0.1$
$\#define\ binf\ =\ 0$
$\#define\ bsup\ =\ \pi$
$\#define\ dx\ =\ abs(bsup - binf)/(precision - 1)$
$\#function\ \ taille(x)\ \ return\ longueur \times max(ymin, sin(x \times dx));$

@axiome@
$\#axiome :\ A(10, 1, 0, 0, 0, < taille(0) >, 10, 10)$

@règles de production@
$p01\ A\{\}\{etape < precision - 1\}\{\} \rightarrow A[A(,,,,, < taille(etape + 1) >,,)]_E$

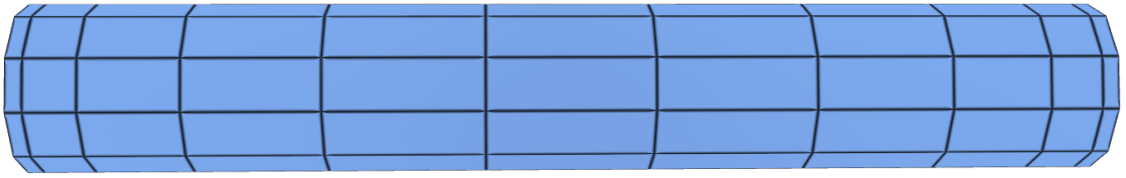


FIGURE 2.6 – Echantillonnage adaptatif. Nous pouvons voir ici le résultat de la grammaire 2.2 pour 10 itérations. Chaque volume généré a une taille définie par f_1.

Dans le système représenté par la grammaire 2.3, la variable $precision$ permet, comme expliqué précédemment, de déterminer le nombre de volumes qui vont former le tronc. La variable $longueur$ permet quant à elle de définir la taille par défaut d'un volume. Ainsi, lorsqu'un volume doit être créé, si il n'est pas influencé par la fonction f_1, alors sa taille sera égale à $longueur$.

Nous avons choisi de prendre les valeurs de la fonction $f_1(x) = sin(x)$ sur $I_1 = [0\,;\pi]$, car c'est sur cet intervalle que la forme de sa courbe correspond à nos besoins. Cependant, pour que cette fonction influe correctement sur la taille de chaque volume, nous devons dans un premier temps échantillonner l'intervalle I_1 en fonction de la précision souhaitée.

Pour ce faire, nous mesurons dans un premier temps la taille de l'intervalle en calculant la distance entre sa borne inférieure $binf$ et sa borne supérieure $bsup$. Puis, nous divisons cette taille par le nombre de volumes du tronc auquel nous en retirons un, car nous voulons que le premier échantillon récupéré dans le système soit $f(0)$. Enfin, le résultat de ce calcul est alors stocké dans la variable dx, qui représente le pas d'échantillonnage selon x.

Une fois que les variables ont été initialisées, nous définissons la fonction $taille(x)$. Cette dernière est utilisée au sein de la règle $p01$ pour définir la taille du volume A créé à chaque itération. Pour ce faire, la fonction calcule dans un premier temps le résultat de $sin(x)$, pour x égal à la valeur de dx multipliée par le nombre d'itérations subies par le système. La valeur ainsi obtenue est ensuite multipliée par la valeur de $longueur$, puis appliquée à la taille du volume. Aussi, pour éviter que la taille ainsi calculée soit nulle, nous introduisons une variable $ymin$ qui permet de fixer la valeur minimale de $sin(x)$ à 0.1 dans notre exemple.

Comme nous pouvons le voir sur la figure 2.6, le tronc ainsi généré respecte les objectifs décrits précédemment. En effet, le volume central est le plus grand volume du tronc, et plus on se rapproche des extrémités de ce dernier, plus la taille des volumes diminue.

Courbure de l'axe central.
La dernière propriété de l'axe central que nous voulions paramétrer est sa courbure. Comme nous avons vu sur la banane de la figure 2.4, la courbure de l'axe central influe directement sur la courbure générale du fruit. Ainsi, pour paramétrer cette propriété, nous avons introduit trois variables de rotation dans notre système : $rotx$, $roty$ et $rotz$. Ces trois variables sont utilisées pour modifier la valeur des attributs de rotation du volume A créé à chaque itération.

Nous noterons qu'il est possible, comme pour l'échantillonnage de l'axe central, d'utiliser des fonctions mathématiques pour faire varier les valeurs de $rotx$, $roty$ et $rotz$ au fil de l'exécution du système. Cependant, nous avons jugé suffisant d'utiliser des angles constants pour créer les résultats présentés dans ce mémoire, permettant ainsi de simplifier l'écriture et la lecture de nos grammaires. Pour générer la banane de la figure 2.4 par exemple, les valeurs des variables de rotations sont $rotx = 0$, $roty = 0$ et $rotz = 10$.

Courbure de la surface du fruit le long du tronc.
Dans la structure arborescente générée par notre système, les branches permettent de dessiner les courbes de la surface du fruit. Elles sont aussi importantes que l'axe central, et influent grandement sur la forme du résultat final. Ainsi, en paramétrant correctement les attributs des branches créées à chaque itération, nous aimerions être capables de modéliser n'importe quelle forme de fruit simple. Pour ce faire, nous avons utilisé une fonction mathématique dont la courbe va définir la forme de la surface que nous voulons obtenir. Elle va ainsi déterminer la taille de chaque branche à ajouter, dans le but de coller à la forme de sa courbe.

Pour définir la forme arrondie de notre fruit, nous avons choisi d'utiliser la fonction $f_2(x) = \sqrt{1-(x-1)^2}$, car cette dernière dessine une courbe en forme de demi-cercle sur l'intervalle $I_2 = [0\,;2]$. L'intégration de cette fonction à un système est montrée dans la grammaire 2.4 et le résultat obtenu est illustré sur la figure 2.7.

Grammaire 2.4 : `Courbure de la surface d'un fruit.`

@définition des volumes@
$\#define\ A(10,1,0,0,0,10,10,10,1,1,1)$
$\#define\ B(4,1,0,0,0,10,2,10,1,1,1)$

@variables générales@
$\#define\ precision = 15$

@paramétrage de la fonction@
$\#define\ longueur = 15$
$\#define\ ymin = 0.1$
$\#define\ binf = -1$
$\#define\ bsup = 1$
$\#define\ dx = abs(bsup - binf)/(precision - 1)$
$\#function\ taille(x)\ return\ longueur \times max(ymin, \sqrt{1-(dx \times x - 1)^2});$

@axiome@
$\#axiome :\ A$

@règles@
$p01\ A\{\}\{etape > 0\ \&\&\ etape < precision\}\{\} \rightarrow A[A]_E$
$p02\ A\{\}\{etape < precision\}\{\} \rightarrow A[B(,,,,,< taille(etape) >,,,,,)]_{C1}$
$p03\ B\{\}\{etape == precision\}\{\} \rightarrow B_{C*}|B_{C*}$

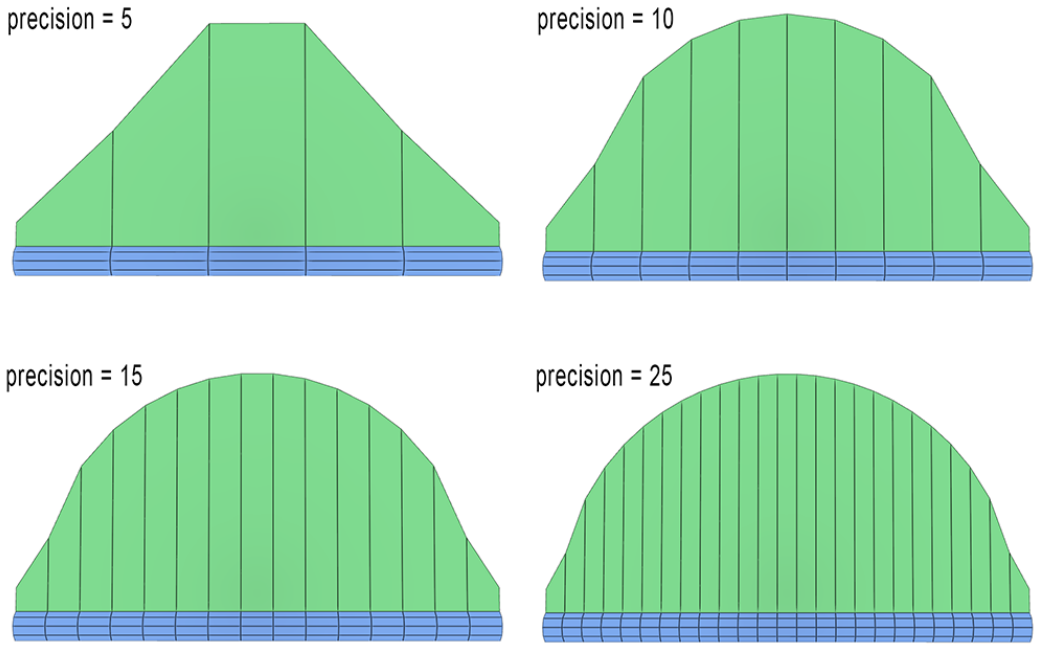


FIGURE 2.7 – Courbure de la surface. Résultats obtenus avec la grammaire 2.4 pour différentes valeurs de la variable $precision$ et pour une taille de tronc fixe.

Le système représenté par la grammaire 2.4 permet de générer une structure arborescente simple, en créant des branches B sur les faces $C1$ d'un tronc composé de volumes A. Une fois que l'arborescence a été créée, les volumes B sont ensuite collés entre eux pour former une courbe pouvant être appliquée à la surface d'un fruit.

Nous ne détaillerons pas ici l'intégration de la fonction $f_2(x) = \sqrt{1-(x-1)^2}$ à notre système, car la méthode utilisée est similaire à celle utilisée pour intégrer la fonction $f_1(x)$ à la grammaire 2.3. La seule différence avec le système précédemment décrit réside dans le fait que la fonction $taille(x)$ permet de déterminer la taille des branches, et non pas la taille des volumes du tronc.

Nous pouvons voir sur la figure 2.7 que la variable $precision$ joue un rôle important dans la qualité de définition de la forme du fruit. En effet, plus le nombre de branches générées sera important, plus la surface du fruit collera parfaitement à la fonction $f_2(x)$.

Courbure de la surface du fruit autour du tronc.
Nous venons de voir que, grâce à la fonction $f_2(x)$, il est possible de faire varier la taille des branches le long du tronc de manière à ce que la surface du fruit corresponde à la forme de sa courbe (figure 2.7). Cependant, pour affiner le contrôle du résultat final, nous voulons pouvoir faire varier la taille d'une branche non seulement en fonction de l'étage du tronc sur lequel elle est générée, mais également en fonction du côté du tronc sur lequel elle se trouve. Pour ce faire, nous allons utiliser une fonction dont l'intervalle sera échantillonné non pas en fonction du nombre de prismes qui forment le tronc, mais en fonction de l'ordre de ces derniers. Ainsi, nous pourrons faire en sorte par exemple que les branches issues de faces $C4$ soient plus grandes que celles qui sont issues de faces $C2$.

Grâce à la paramétrisation du tour du tronc, nous aimerions pouvoir créer des bosses autour de nos modèles afin de générer une variété spécifique de tomates : les tomates côtelées. La figure 2.8 montre deux exemples de tomates côtelées que nous avons obtenu grâce à notre 3G-carte L-système final, présenté plus loin dans ce mémoire. Pour pouvoir représenter les bosses (ou protubérances) autour du tronc, nous allons utiliser la fonction $f_4(x) = |sin(x)|$ échantillonnée sur un intervalle qui va changer en fonction du nombre de protubérances désirées. L'intégration de cette fonction à un système est montrée dans la grammaire 2.5.

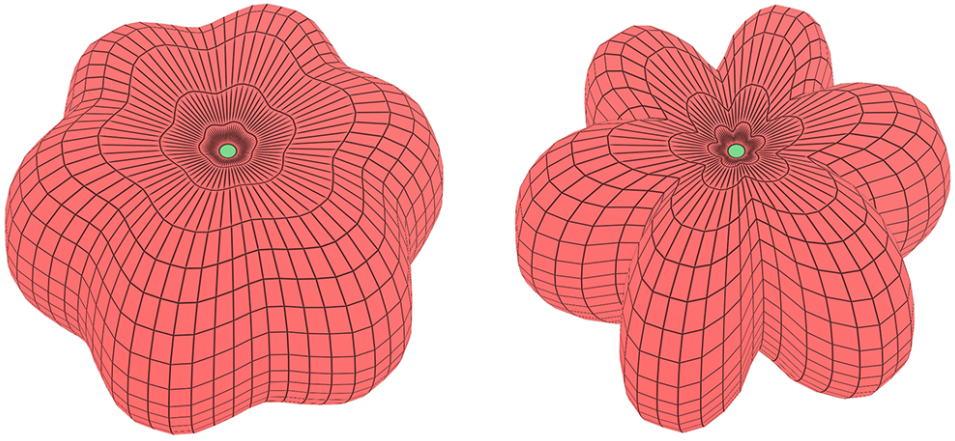


FIGURE 2.8 – Exemple de tomates à côtes.

Grammaire 2.5 : `Courbure de la surface du fruit autour du tronc`

@définition des volumes@
$\#define\ A(10,1,0,0,0,10,10,10)$
$\#define\ B(4,1,0,0,0,10,2,10)$

@variables générales@
$\#define\ ph\ =\ 11$

@paramétrage de la fonction@
$\#define\ creux\ =\ 0.5$
$\#define\ nbprotus\ =\ 6$
$\#define\ longueur\ =\ 50$
$\#define\ ymin\ =\ 0.1$
$\#define\ binf\ =\ 0$
$\#define\ bsup\ =\ \pi \times nbprotus$
$\#define\ dx\ =\ abs(bsup - binf)/(ph-1)$
$\#function\ \ taille(x)\ \ return\ longueur \times max(ymin, abs(sin(x \times dx)));$

@axiome@
$\#axiome :\ A(< ph >, 1, 0, 0, 0, 10, 10, 10)$

@règles de production@
$p01\ A\{\}\{etape == 0\}\{\} \rightarrow A[B(,,,,,< taille(numface-1) >,,,,,)]_{C*}$

$p02\ B\{\}\{etape == 0\}\{\} \rightarrow B_{C*}|B_{C*}$

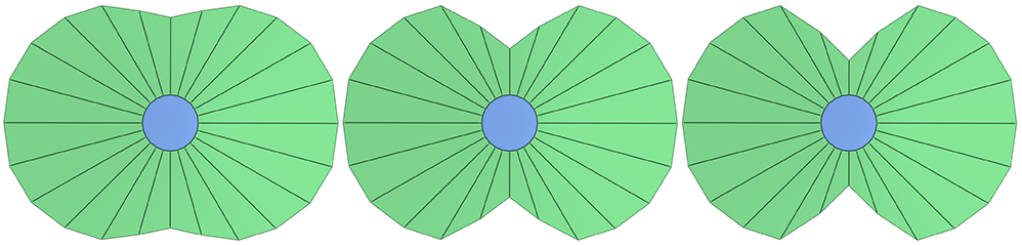


FIGURE 2.9 – De gauche à droite : résultats obtenus à l'aide du système de la grammaire 2.5 pour $creux$ égale à 0.9, 0.5 et 0.2.

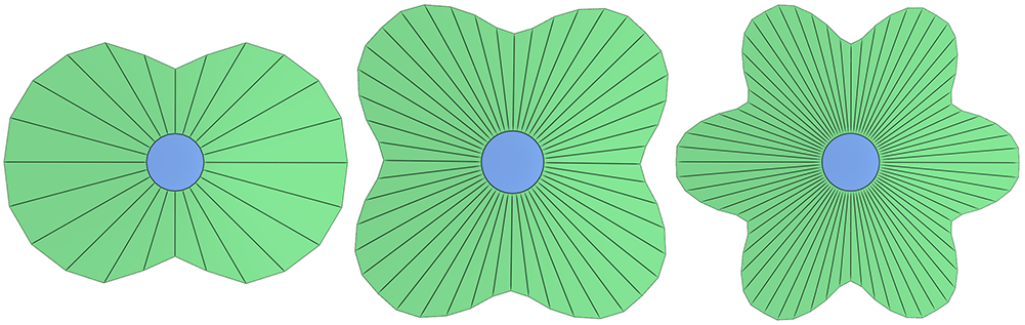


FIGURE 2.10 – De gauche à droite : résultats obtenus à l'aide du système de la grammaire 2.5 pour $nbprotus$ égale à 2, 4 et 6.

Le système représenté par la grammaire 2.5 permet de générer un unique volume A, sur lequel des branches B vont être créées puis collées entre elles. Dans cet exemple, la variable ph représente la précision horizontale du tronc, c'est-à-dire l'ordre des volumes A du système. Quant aux variables $nbprotus$ et $creux$, elles permettent de définir respectivement le nombre de bosses (ou protubérances) formées par les branches autour du tronc et la profondeur des creux qui se trouvent entre ces bosses. Le reste des variables remplissent les mêmes fonctions que celles qui portent le même nom dans la grammaire 2.4.

Dans cette grammaire, nous introduisons également une nouvelle variable globale appelée $numface$ et qui permet de récupérer le numéro de la face courante sur laquelle une règle est appliquée. Pour des faces extrémités ou origines, cette variable sera égale à 0, alors que pour des faces latérales, sa valeur correspondra aux numéros de ces dernières ($numface = 1$ pour une face $C1$, $numface = 2$ pour une face $C2$, etc...).

Dans cet exemple, nous avons choisi d'utiliser la fonction $f_4(x) = |sin(x)|$ échantillonnée sur l'intervalle $I_4 = [0\,;\pi \times nbprotus]$ car sa courbe dessine un nombre de bosses égal à la valeur de la variable $nbprotus$. Ici, contrairement aux autres intervalles présentés tout au long de cette section, l'intervalle I_4 ne sera pas échantillonné en fonction du nombre de volumes qui forment le tronc, mais en fonction de l'ordre de ces derniers. Les figures 2.9 et 2.10 illustrent l'influence des variables $creux$ et $nbprotus$ sur le résultat des itérations du système représenté par la grammaire 2.5.

Angle des branches par rapport au tronc.
Chez certains fruits, il est possible d'observer des creux au niveau leurs extrémités. Chez la tomate par exemple, nous pouvons remarquer la présence d'un creux au niveau de son pédoncule (voir figure 2.2). Dans la grammaire générique présentée au début de ce chapitre (gramaire 2.1), pour représenter ce creux chez la tomate, nous avons fait varier les angles de certaines branches par rapport au tronc. Ainsi, les branches qui se trouvent au niveau du pédoncule ont un attribut de rotation atz de valeur inférieure à ceux des autres branches (-25 degrés dans la grammaire 2.1). Et plus on se rapproche du centre, plus la valeur de la variable atz des branches augmente, pour arriver à une valeur maximum de 0 degré, formant ainsi des branches qui sont perpendiculaires au tronc.

Dans le but de pouvoir représenter des creux au niveau des extrémités du fruit, nous avons utilisé une fonction mathématique qui nous permet de faire une transition linéaire entre un angle minimum et un angle maximum définis par l'utilisateur. Comme pour les autres grammaires présentées dans cette section, l'intervalle de cette fonction sera échantillonné en fonction du nombre de volumes sur lesquels nous voulons qu'elle influe. Cependant, contrairement aux autres fonctions, plutôt que d'influencer toutes les branches de l'arborescence, celle-ci a la particularité de pouvoir cibler qu'une partie des branches. La grammaire 2.6 nous permet de voir comment intégrer cette fonction dans un système, et la figure 2.11 illustre le résultat de l'exécution de système.

Grammaire 2.6 : `Angles des branches par rapport au tronc.`

@définition des volumes@
$\#define\ A(10,1,0,0,0,10,5,5,1,1,1)$
$\#define\ B(4,1,0,0,0,50,2,10,1,1,1)$

@variables générales@
$\#define\ precision = 10$

@paramétrage de la fonction@
$\#define\ binf = -50$
$\#define\ bsup = -10$
$\#define\ nbvol = 5$
$\#define\ dx = abs(bsup - binf)/(nbvol - 1)$
$\#function\ angle(x)\ return\ bsup - (x - (precision - nbvol)) * dx;$

@axiome@
$\#axiome :\ A$

@règles@
$p01\ A\{\}\{etape > 0\ \&\&\ etape < precision\}\{\} \rightarrow A[A]_E$

$p02\ A\{\}\{etape < precision - nbvol\}\{\} \rightarrow A[B]_{C*}$
$p03\ A\{\}\{etape \geq precision - nbvol\}\{\} \rightarrow A[B(,,,,< angle(etape) >,,,,,,)]_{C*}$

$p04\ B\{\}\{etape == precision\}\{\} \rightarrow B_{C*}|B_{C*}$

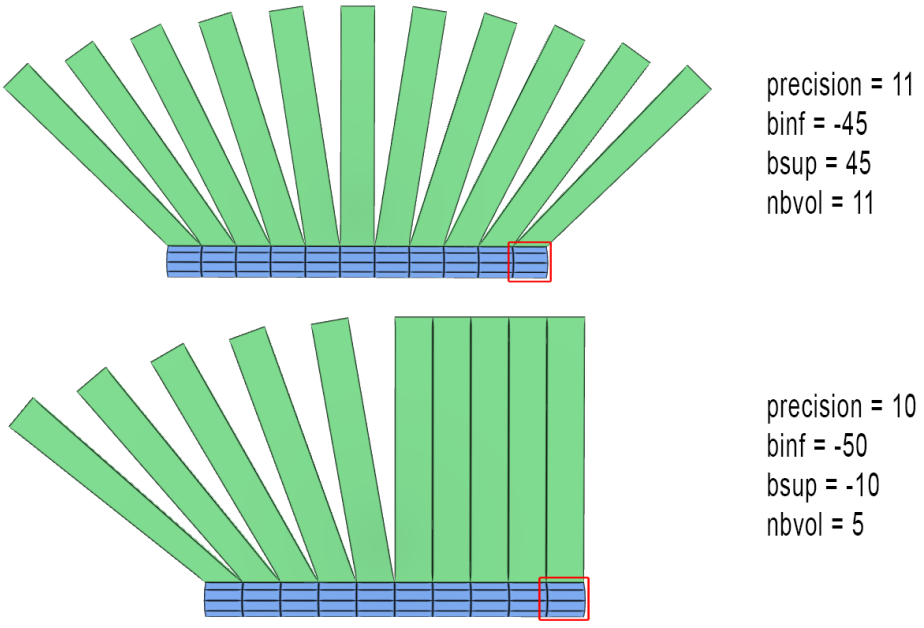


FIGURE 2.11 – Angles des branches par rapport au tronc. Deux résultats différents obtenus avec l'utilisation de valeurs différentes pour les variables de la grammaire 2.6. L'axiome est encadré en rouge sur chaque résultat.

Le système représenté par la grammaire 2.6 permet de générer une structure arborescente composée d'un tronc de volumes A sur lequel des branches B sont créées, puis collées entre elles.

Comme dans la grammaire 2.4, une fonction mathématique va permettre d'agir sur les attributs des branches qui sont créées à chaque itération. Dans cet exemple, la fonction mathématique $f_4(x) = bsup - x$ définie sur l'intervalle $I_4 = [binf\,;bsup]$ permet de définir l'attribut de rotation atz des branches qui sont créées à chaque itération. Cette fonction permet de faire une transition linéaire entre un angle maximum, représenté par la variable $bsup$, et un angle minimum représenté par $binf$.

Pour influencer correctement les angles des branches, l'intervalle I_4 de la fonction $f_4(x)$ est échantillonné en fonction d'une précision définie, tout comme les intervalles I_1 et I_2. Cependant, la différence avec ces deux derniers réside dans le fait que I_4 n'est pas échantillonné par rapport au nombre de volumes A de la structure arborescente (égal à la valeur de $precision$), mais par rapport à une variable spéciale appelée $nbvol$. Cette dernière permet de définir le nombre d'étages du tronc, depuis le pédoncule, dont les branches vont être modifiées par f_4. Ainsi, dans la grammaire 2.6 par exemple, la règle $p03$ ne sera pas appliquée tant que le nombre de volumes A qu'il reste à générer n'est pas égal à $nbvol$. Ceci a pour conséquence, comme nous pouvons le voir en bas de la figure 2.11 par exemple, de ne modifier que les branches qui se trouvent en haut de la structure arborescente. De plus, si l'utilisateur souhaite créer des creux aux deux extrémités du fruit, il lui suffit de définir une valeur de $nbvol$ égale au nombre de volumes A (voir figure 2.11).

2.3.3 Synthèse et grammaire finale.

Dans cette section nous avons introduit une méthode permettant de généraliser la forme des fruits simples charnus. En effet, en partant de l'observation que ces fruits présentent tous une symétrie rotationnelle autour d'un axe, nous proposons de générer leur forme à l'aide d'un 3G-carte L-système qui crée une structure similaire à celle d'un arbre. L'axe de symétrie rotationnel y est alors représenté par un ensemble de prismes formant un "tronc", qui croît au fil des itérations, et autour duquel des branches sont créées pour représenter la paroi du fruit.

Une fois que la structure générale du système a été mise en place, nous avons introduit une version paramétrique de notre modèle. Pour ce faire, nous avons proposé un moyen de décomposer la forme relativement complexe d'un fruit en plusieurs formes simples. Ces dernières, représentées par des fonction mathématiques classiques (exemple : la fonction sinus), nous ont ainsi permis, une fois combinées entre elles, de créer un 3G-carte L-système paramétrique complet qui est capable de générer n'importe quelle forme de fruit simple charnu à l'aide de paramètres simples. Ce système est représenté par la grammaire 2.7, et nous montrons sur les figures 2.13 et 2.14 un ensemble de résultats qu'il est possible d'obtenir en utilisant les valeurs de paramètres définis dans le tableau de la figure 2.12.

Grammaire 2.7 : `3G-carte L-système paramétrique de fruit simple.`

@Définition des volumes@
$\#define\ A(10,1,0,0,0,1,5,5)$
$\#define\ B(4,1,0,0,0,70,2,5)$

@variables générales@
$\#define\ precision = 10$
$\#define\ ph\ =\ 11$
$\#define\ largeur = 55$
$\#define\ longueur = 15$
$\#define\ creux\ =\ 0.5$
$\#define\ nbprotus\ =\ 6$
$\#define\ anglesup = 0$
$\#define\ angleinf = -30$
$\#define\ nbvolangle = 5$

@fonction d'échantillonnage adaptatif du tronc@
$\#define\ binfc = 0$
$\#define\ bsupc = \pi$
$\#define\ lmin = 0.1$
$\#define\ dxc = abs(bsupc - binfc)/(precision - 1)$
$\#function\ fc(x)\ return\ longueur \times max(lmin, sin(x \times dxc + binfc));$

@courbure du tronc@
$\#define\ rotx = 0$
$\#define\ roty = 0$
$\#define\ rotz = 0$

@taille des branches en fonction de leur étage dans le tronc@
$\#define\ bsupb = 1$
$\#define\ binfb = -1$
$\#define\ wminb = 0.1$
$\#define\ dxb = abs(bsupb - binfb)/(precision - 1)$
$\#function\ fb(x)\ return\ max(wminb, \sqrt{1 - (dxb \times x + binfb)^2});$

@taille des branches en fonction de la face génératrice@
$\#define\ wmind\ =\ 0.1$
$\#define\ binfd\ =\ 0$
$\#define\ bsupd\ =\ \pi \times nbprotus$
$\#define\ dxd\ =\ abs(bsupd - binfd)/(ph - 1)$
$\#function\quad fd(x)\ \ return\ max(wmind, abs(sin(x \times dx)));$

@fonction d'angle de branches@
$\#define\ dxa = abs(anglesup - angleinf)/(nbvolangle - 1)$
$\#function\ fa(x)\ return\ anglesup - (x - (precision - nbvolangle)) \times dxa;$

@définition de l'axiome de départ@
$\#axiome : A$

@règle de croissance du tronc@
$p01\ A\{\}\{etape > 0\ \&\&\ etape < precision\}\{\} \rightarrow A[A(,,< rotx >,< roty >, < rotz >,< fc(etape) >,< fc(etape) >,< fc(etape) >)]_E$

@règles de croissance des branches@
$p02\ A\{\}\{etape < precision - nbvolangle\}\{\} \rightarrow A[B(,,,,,< largeur \times fb(etape) \times fd(numface - 1) >,,< fc(etape) >)]_{C*}$

$p03\ A\{\}\{etape \geq precision - nbvolangle\}\{\} \rightarrow A[B(,,,,< fa(etape) >,< largeur \times fb(etape) \times fd(numface - 1) >,,< fc(etape) >)]_{C*}$

@règle de collage des branches@
$p04\ B\{\}\{etape == precision\}\{\} \rightarrow B_{C*}|B_{C*}$

Fruit	Précision	Courbure du tronc	Échantillonnage du tronc	Taille des branches le long du tronc	Taille des branches autour du tronc	Angle des branches
Tomate ronde	10	$rotx = 0$ $roty = 0$ $rotz = 0$	$longueur = 13$ $f(x) = \sin(x)$ $I = [\,0\,;\,\pi\,]$	$largeur = 55$ $f(x) = \sqrt{1-x^2}$ $I = [\,-1\,;1\,]$	$nbprotus = 0$ $creux = 0$	$nbvol = 5$ $I = [\,-60\,;0\,]$
Banane	10	$rotx = 0$ $roty = 0$ $rotz = 12$	$longueur = 80$ $f(x) = longeur$ $I = [\,0\,;1\,]$	$largeur = 90$ $f(x) = \sqrt{1-x^2}$ $I = [\,-1\,;1\,]$	$nbprotus = 0$ $creux = 0$	$nbvol = 0$ $I = [0\,;0\,]$
Tomate cerise	20	$rotx = 0$ $roty = 0$ $rotz = 0$	$longueur = 10{,}5$ $f(x) = longeur$ $I = [\,0\,;1\,]$	$largeur = 100$ $f(x) = \sqrt{1{,}5 * (1-x^2)}$ $I = [\,-1\,;1\,]$	$nbprotus = 0$ $creux = 0$	$nbvol = 15$ $I = [\,-15\,;0\,]$
Poire	20	$rotx = 0$ $roty = 0$ $rotz = 0$	$longueur = 17$ $f(x) = x^3(1{,}5-x) + 0.46$ $I = [\,-0{,}6\,;1{,}6\,]$	largeur = 60 $f(x) = x^3(1{,}5-x) + 0.46$ $I = [\,-0{,}6\,;1{,}6\,]$	$nbprotus = 0$ $creux = 0$	$nbvol = 0$ $I = [0\,;0\,]$
Tomate allongée	20	$rotx = 0$ $roty = 0$ $rotz = 0$	$longueur = 25$ $f(x) = \sin(x)$ $I = [\,0\,;\,\pi\,]$	$largeur = 60$ $f(x) = \sqrt{1-x^2}$ $I = [\,-1\,;1\,]$	$nbprotus = 0$ $creux = 0$	$nbvol = 15$ $I = [\,-60\,;0\,]$
Tomate côtelée	20	$rotx = 0$ $roty = 0$ $rotz = 0$	$longueur = 5$ $f(x) = \sin(x)$ $I = [\,0\,;\,\pi\,]$	$largeur = 55$ $f(x) = \sqrt{1-x^2}$ $I = [\,-1\,;1\,]$	$nbprotus = 6$ $creux = 0{,}5$	$nbvol = 15$ $I = [\,-60\,;0\,]$

FIGURE 2.12 – Paramètres requis pour dessiner les modèles de fruits.

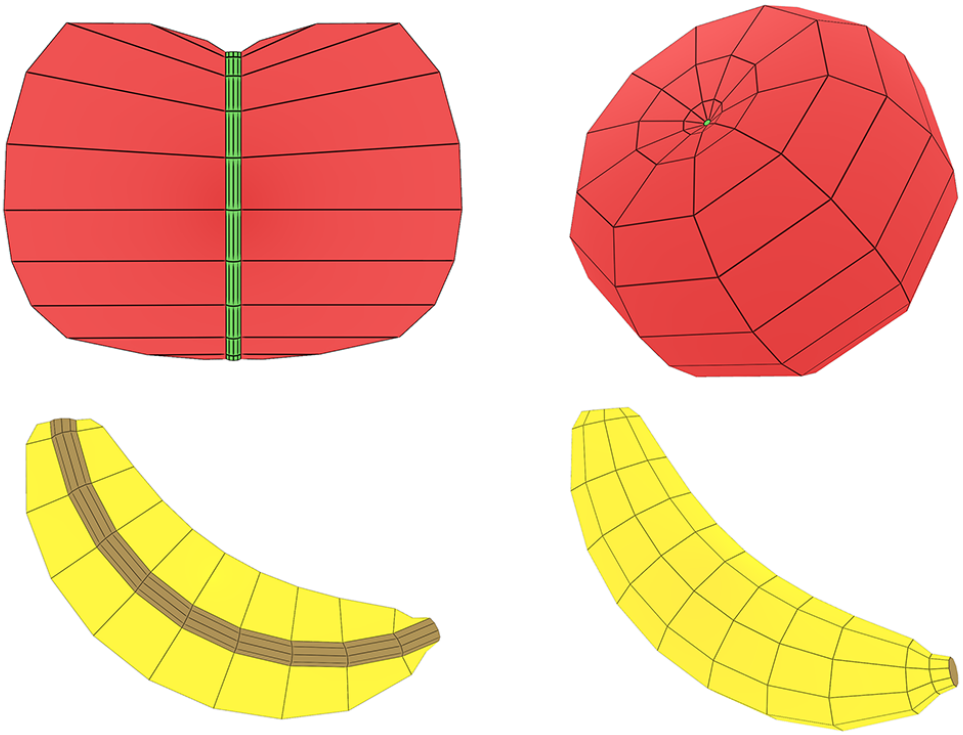


FIGURE 2.13 – Résultats obtenus à l'aide des paramètres du tableau 2.12.

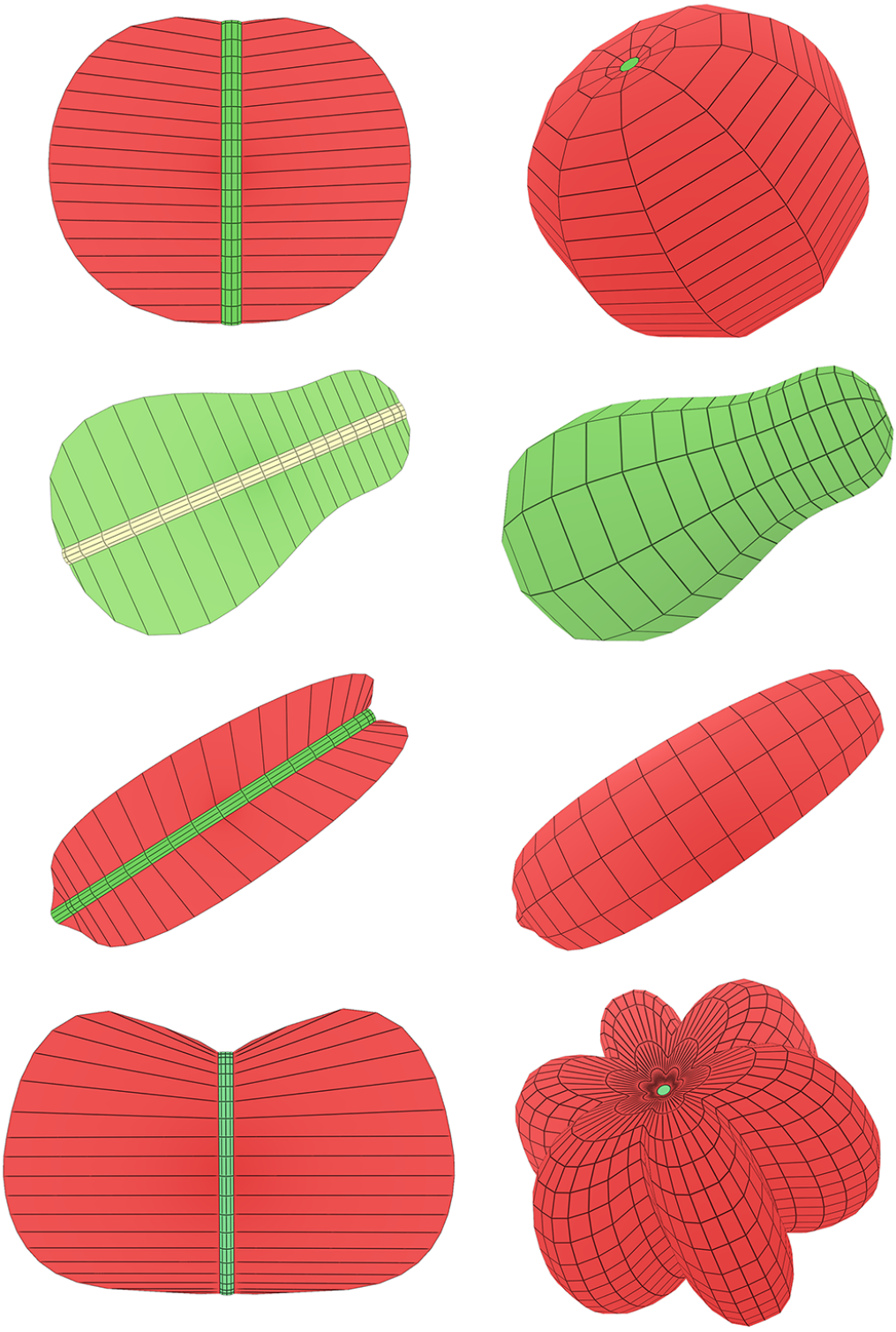


FIGURE 2.14 – Résultats obtenus à l'aide des paramètres du tableau 2.12.

Comme nous pouvons le voir sur les figures 2.13 et 2.14, notre 3G-carte L-système paramétrique permet d'obtenir plusieurs formes de fruits différentes. Pour ce faire, il suffit de choisir correctement les fonctions mathématiques chargées d'influencer les différentes caractéristiques de la structure arborescente générée par notre système. De plus, grâce à un ensemble de variables ciblant des caractéristiques précises du fruit, il est possible de faire varier la forme globale de ce dernier de manière relativement simple. Nous pouvons voir par exemple sur les figures 2.15 à 2.21 l'influence de quelques unes de ces variables sur la forme générale du fruit. Bien sûr les formes obtenues ne sont pas toujours réalistes, et l'utilisateur devra ajuster chacune des variables du système de manière à obtenir une forme plus ou moins plausible. De plus, pour aller plus loin, l'utilisateur pourra également intégrer de l'aléatoire aux fonctions du système de manière à ce que le résultat final ait une forme différente à chaque exécution.

L'une des autres forces de notre méthode réside dans le fait que, bien que nous proposons un ensemble de fonctions et de variables bien spécifiques pour répondre à notre problématique, notre système a été créé de manière à pouvoir les remplacer de manière relativement simple dans le cas où l'utilisateur aurait besoin de générer des formes différentes. Nous noterons cependant deux principaux défauts à notre système représenté par la grammaire 2.7. Premièrement, lorsque le nombre de volumes utilisés pour générer le fruit n'est pas suffisant, ce dernier a tendance à avoir une forme qui n'est pas assez organique, pas assez lisse. Le second défaut réside dans le fait que notre système actuel ne se charge que de créer la surface du fruit sans se soucier de sa structure interne. Nous présenterons dans la section suivante une extension que nous avons introduit au système, permettant de pallier ces deux limitations.

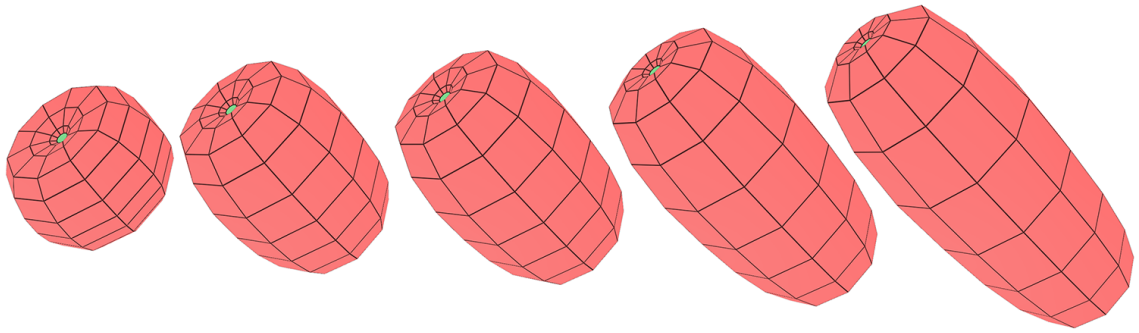


FIGURE 2.15 – Augmentation de la valeur de $longueur$ dans la grammaire 2.7.

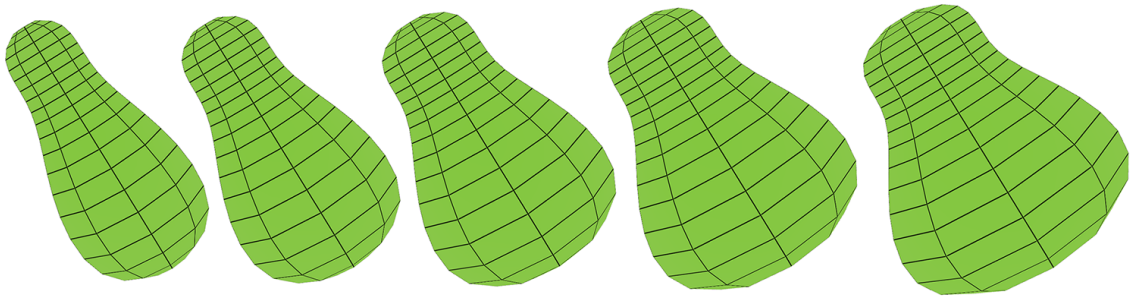


FIGURE 2.16 – Augmentation de la valeur de $largeur$ dans la grammaire 2.7.

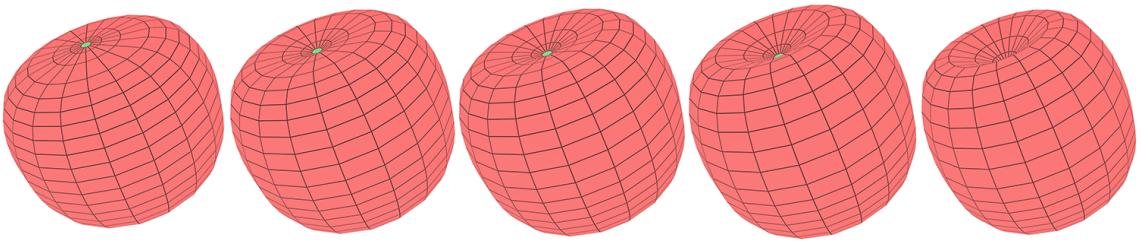


FIGURE 2.17 – Augmentation de l'angle des branches pour créer un creux de plus en plus profond au niveau du pédoncule.

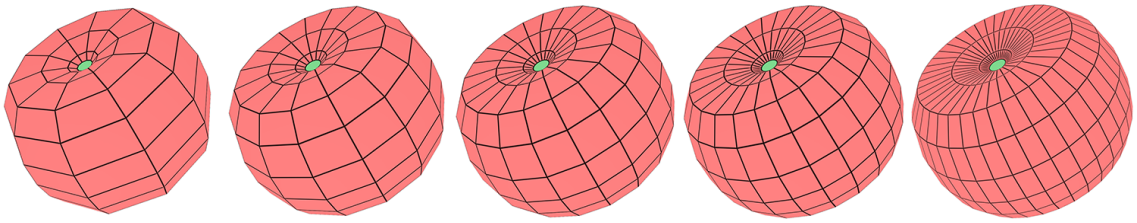


FIGURE 2.18 – Augmentation de la précision horizontale pour avoir plus de branches autour du tronc et dessiner de manière plus précise les protubérances de la tomate.

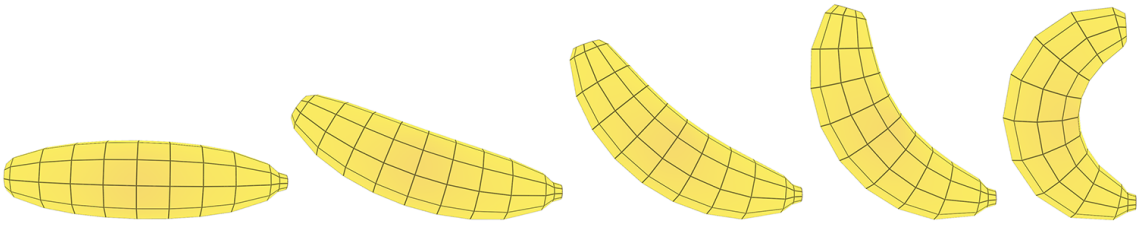


FIGURE 2.19 – Augmentation de l'angle de courbure de l'axe central.

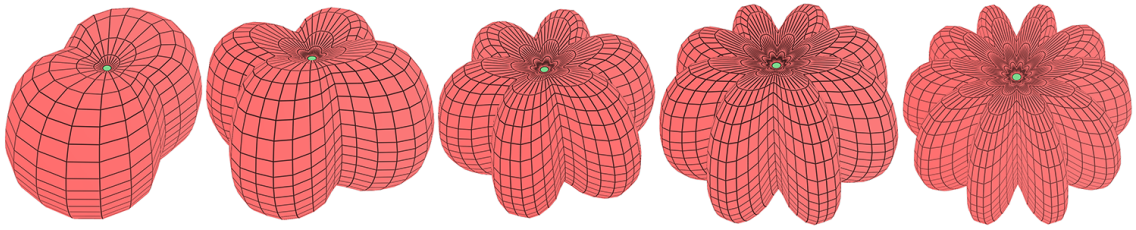


FIGURE 2.20 – Augmentation du nombre de protubérances sur une tomate côtelée.

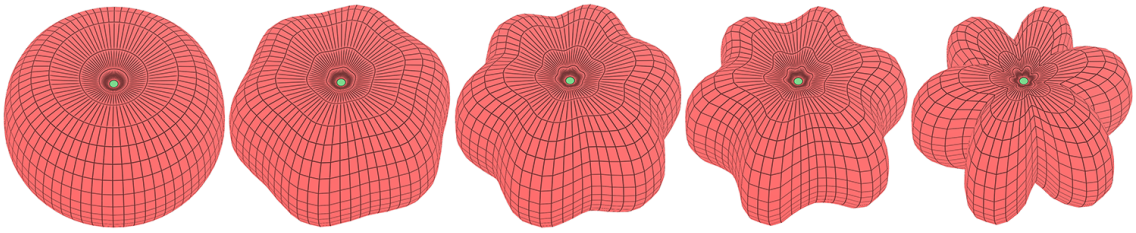


FIGURE 2.21 – Augmentation du creux entre les protubérances.

2.4 Modélisation de la structure interne des fruits

Maintenant qu'il est possible de modéliser correctement la surface d'un fruit à l'aide de notre 3G-carte L-système, nous voulons pouvoir générer sa structure interne de manière relativement simple. Pour ce faire, nous devons être capables de représenter non seulement les différentes couches du péricarpe d'un fruit, mais aussi ses différents carpelles et ses graines. Dans cette section, nous présenterons dans un premier temps comment utiliser les outils que nous avons à notre disposition pour représenter les différentes couches du péricarpe. Ensuite, nous introduirons une extension basée sur les subdivisions volumiques et surfaciques. Cette extension permet de créer des volumes de forme arrondie, ce qui facilite la représentation de graines. Enfin, nous montrerons comment nous avons utilisé cette extension pour introduire de nouvelles règles de production qui ont facilité la génération des différents éléments de la structure interne d'un fruit.

2.4.1 Représentation des couches du péricarpe

Un premier pas vers la génération de la structure interne d'un fruit consiste à représenter les trois couches de son péricarpe : l'exocarpe, le mésocarpe et l'endocarpe. Pour ce faire, deux approches s'offrent à nous. La première, illustrée sur la figure 2.22, consiste à ajouter ces couches directement sur la surface du fruit qui a été créée avec notre système paramétrique. Cette méthode, relativement simple à mettre en place, va ainsi ajouter de nouvelles branches sur celles qui ont déjà été créées. La deuxième méthode quant à elle, présentée en détails dans la suite de ce mémoire, consiste à subdiviser la structure volumique dans le but de créer les couches à l'intérieur même de la surface. Par rapport à la première approche, celle-ci présente l'avantage de ne pas influencer la forme de la surface qui a déjà été mise en place par notre système.

La figure 2.22 illustre la première approche appliquée à une demi-sphère générée par notre système. Dans cet exemple, la demi-sphère représente l'endocarpe du fruit. Ainsi, une fois que ce dernier a été correctement généré, nous commençons par ajouter une nouvelle branche C sur l'extrémité de chaque branche B du modèle. Les branches C ainsi créées sont ensuite collées entre elles pour former le mésocarpe. Une fois cette étape terminée, une nouvelle règle se charge d'ajouter des branches D sur les extrémités des branches C. Enfin, les branches D sont collées entre elles pour former l'exocarpe. Le modèle ainsi créé peut être vu comme la coupe d'un fruit simple composé d'un seul carpelle.

La grammaire 2.8 est une version simplifiée de celle qui génère le fruit de la figure 2.22. En effet, pour faciliter la lecture, nous n'avons pas représenté dans cette grammaire les différentes parties qui sont chargées de générer la surface du fruit, car elles ont déjà été présentées au début de ce chapitre. Comme nous pouvons le voir dans cet exemple, la première approche est relativement simple à mettre en place dans notre système. En effet, il suffit d'ajouter deux règles d'ajout ($p05$ et $p07$) et deux règles de collage ($p06$ et $p08$) à notre grammaire pour que le mésocarpe et l'endocarpe du fruit soient générés correctement. L'inconvénient cependant réside dans le fait que, dans certains cas, la génération des couches sur la surface déjà créée peut engendrer une déformation de cette dernière. Dans ces cas là, l'ajustement de la forme devient alors une tâche très fastidieuse et an-

nule tous les avantages de l'utilisation d'un système paramétrique pour la génération de la forme du fruit. Nous présenterons dans les sections suivantes l'extension que nous avons introduit pour pallier cette limitation.

Grammaire 2.8 : `Génération simple des couches du péricarpe.`

@définition des volumes A et B@

...

$\#define\ C(4,1,0,0,0,10,2,2)$

$\#define\ D(4,1,0,0,0,5,2,2)$

@définition des variables et de l'axiome@

...

@création de la surface du fruit@

...

@ajout du mésocarpe et de l'exocarpe@

$p05\ B\{\}\{etape == precision + 1\}\{\} \rightarrow B[C]_E$

$p06\ C\{\}\{etape == precision + 1\}\{\} \rightarrow C_{C*}|C_{C*}$

$p07\ C\{\}\{etape == precision + 1\}\{\} \rightarrow C[D]_E$

$p08\ D\{\}\{etape == precision + 1\}\{\} \rightarrow D_{C*}|D_{C*}$

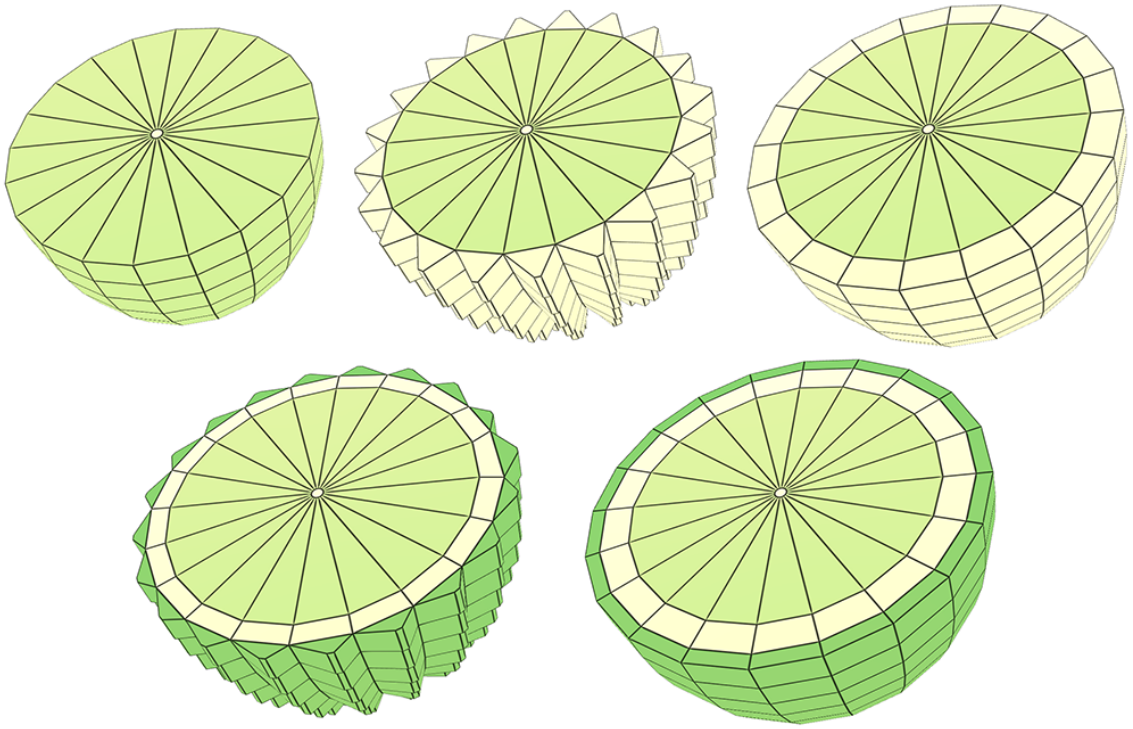


FIGURE 2.22 – Résultat de l'application de la grammaire 2.8.

2.4.2 Intégration de la subdivision aux 3G-cartes L-systèmes

Lorsque nous travaillons avec des 3G-cartes L-systèmes, les volumes utilisés pour générer notre structure arborescente sont des prismes. Comme nous l'avons vu dans le chapitre précédent, cette catégorie de volumes présente l'avantage principal de pouvoir introduire une notion d'orientation dans le modèle. Cependant, bien que très utiles pour générer la surface d'un fruit, les prismes présentent un inconvénient majeur lorsqu'il s'agit de modéliser sa structure interne. En effet, ils ne nous donnent pas la possibilité de créer des volumes ayant une forme plus arrondie, et donc plus organique, comme des graines par exemple. Ainsi, en plus de pouvoir générer les différentes couches du péricarpe sans influencer la forme déjà obtenue par notre système, nous voulons être capables de générer des volumes ronds pour représenter des graines dans nos fruits.

Pour répondre à ces deux problématiques, nous avons introduit l'utilisation de la subdivision de Catmull-Clark [CC98] au concept des 3G-cartes L-systèmes. En effet, parmi tous les schémas de subdivision existant, nous avons identifié ce dernier comme étant le plus adapté à nos besoins pour deux raisons principales. Premièrement, l'application de ce schéma donne des maillages polygonaux de bonne qualité après très peu d'itérations. Et deuxièmement, lorsque nous l'appliquons à des maillages quelconques, ce schéma a pour particularité de toujours générer des maillages quadrilatéraux, un type de maillage qui est facilement adaptable aux 3G-cartes L-systèmes. La subdivision de Catmull-Clark étant très connue en informatique graphique, nous n'aborderons pas son aspect technique dans ce mémoire. Pour plus de détails sur l'implémentation de cette méthode, nous invitons le lecteur à se référer à [CC98].

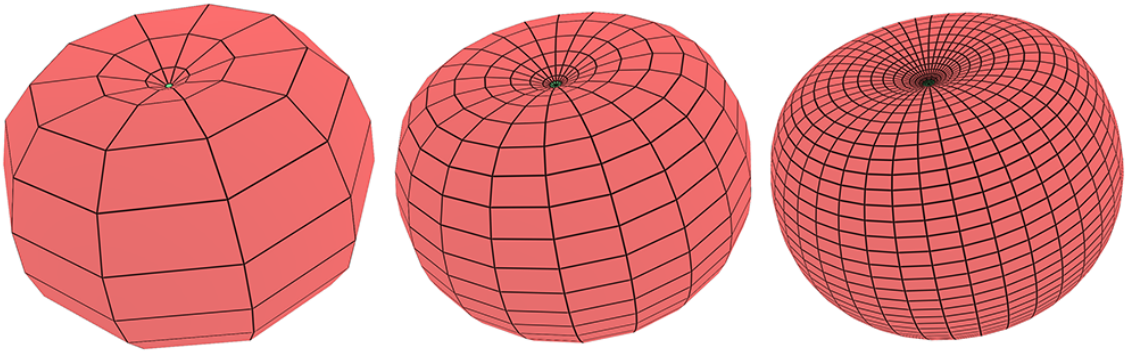


FIGURE 2.23 – De gauche à droite : une tomate à laquelle nous appliquons zéro, une et deux subdivisions de Catmull-Clark.

2.4.2.1 Subdivisions surfacique et volumique

Afin de répondre au mieux aux deux problématiques citées au début de cette section, nous avons introduit dans le concept des 3G-cartes L-systèmes deux extensions basées sur la subdivision de Catmull-Clark : la subdivision surfacique et la subdivision volumique.

Subdivision surfacique.
Ce type de subdivision agit directement, comme son nom l'indique, sur la surface des volumes. Lorsqu'on subdivise un prisme dans une 3G-carte, chacune de ses faces sera alors remplacée par quatre sous-faces ayant le même label que leur face mère, et les coordonnées des nouveaux sommets vont être calculées en utilisant les formules décrites dans [CC98]. Les étapes de ce procédé son illustrées sur la figure 2.24.

Afin de pouvoir utiliser la subdivision surfacique dans un 3G-carte L-système, nous avons introduit dans notre modèle une nouvelle règle de production. Cette dernière permet d'appliquer une ou plusieurs subdivisions surfaciques à des volumes spécifiques ou au modèle tout entier. Ainsi, la règle $A \rightarrow [A]\#n$ appliquera n subdivisions à tous les volumes A du modèle, alors que la règle $* \rightarrow [*]\#n$ subdivisera la surface du modèle entier n fois.

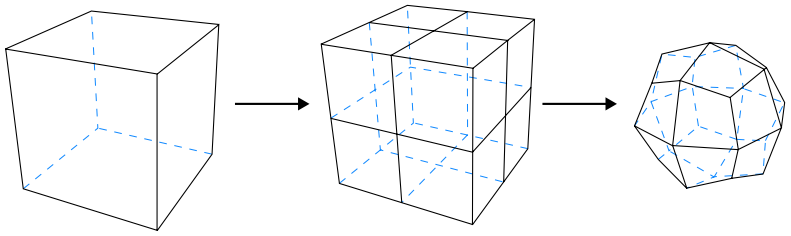


FIGURE 2.24 – Subdivision surfacique. Les faces du prisme sont remplacées par quatre sous-faces, puis on déplace les nouveaux sommets selon le schéma de Catmull-Clark.

Subdivision volumique.
Contrairement à la subdivision surfacique, la subdivision volumique agit aussi bien sur la surface d'un prisme que sur son intérieur. Si nous l'appliquons à un volume, celui-ci sera alors dans un premier temps éclaté en plusieurs volumes fils d'ordre 4. Ensuite, les sommets formant la surface de la structure de volumes ainsi obtenue seront déplacés selon les formules de [CC98]. Comme nous pouvons le voir sur les figures 2.24 et 2.25, lorsque nous utilisons une subdivision volumique à la place d'une subdivision surfacique, le résultat visuel ne change pas. La seule différence réside dans le fait que le résultat final n'est pas composé que d'un seul volume, mais d'une structure de volumes.

Tout comme pour la subdivision de surfaces, nous avons introduit dans notre modèle une nouvelle règle de production permettant d'appliquer la subdivision volumique. Ainsi, si l'utilisateur souhaite appliquer n subdivisions volumiques aux volumes A d'un système, il utilisera la alors la règle $A \rightarrow [A]\#\#n$. Si il souhaite au contraire appliquer n subdivisions volumiques à tout le modèle, alors il utilisera la règle de production $* \rightarrow [*]\#\#n$.

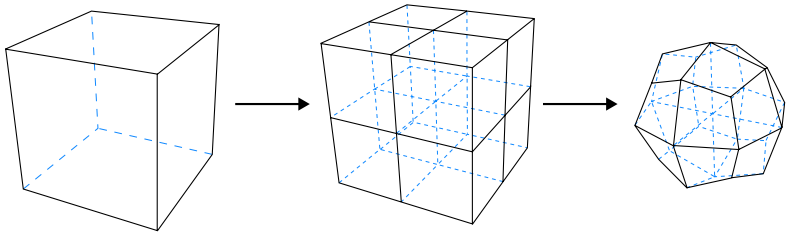


FIGURE 2.25 – Subdivision volumique. Le prisme est éclaté en quatre prismes fils d'ordre 4, puis les sommets de la surface sont déplacés selon le schéma de Catmull-Clark.

2.4.2.2 Comparaison des deux extensions

Chacun des deux types de subdivision présente ses propres avantages et inconvénients, et l'utilisateur devra choisir celui qui lui convient en fonction de ses besoins. Afin de mesurer les performances de ces deux extensions, nous avons d'abord effectué des mesures sur un système permettant de générer une tomate ronde sans la subdiviser (figure 2.23). Ensuite, nous avons effectué les mêmes mesures sur un système permettant cette fois-ci de générer la même tomate à laquelle deux subdivisions surfaciques sont appliquées. Et enfin, des mesures ont été faites sur un dernier système permettant de générer la tomate à laquelle deux subdivisions volumiques sont appliquées. Les mesures ainsi obtenues ont été répertoriées dans le tableau de la figure 2.26.

Tomate ronde	Nombre de brins	Nombre de sommets	Nombre de faces	Nombre de volumes	Temps de génération (en ms)
Aucune subdivision	6 000	220	720	110	422
Deux subdivisions surfaciques	96 000	5 731	12 000	110	1 995
Deux subdivisions volumiques	384 000	9 061	48 000	8 000	3 689

FIGURE 2.26 – Tableau illustrant la structure de données représentée par les tomates de la figure 2.23 ainsi que les temps de génération correspondants. Ces tests ont été faits sur un ordinateur ayant la configuration suivante : une carte graphique Nvidia GeForce GTX690, un processeur Intel Core i7-3770 (3.40GHz) et 16Go de mémoire vive.

En nous basant sur le tableau de la figure 2.26, nous pouvons voir que par rapport à la subdivision volumique, la subdivision surfacique présente l'avantage de ne pas surcharger le système avec un grand nombre d'informations à gérer. En effet, contrairement à la subdivision volumique qui crée des nouveaux volumes, nous ne créons ici que des nouvelles faces et des nouveaux sommets. Ainsi, nous pouvons voir sur le tableau de la figure 2.26 que, pour un même modèle de tomate ronde, l'application de deux subdivisions surfaciques générera quatre fois moins de brins que l'application de deux subdivisions volumiques. De plus, le fait que nous générons moins de nouveaux éléments avec cette méthode se ressent également dans le temps que prend le système pour générer le modèle final. En effet, la génération prendra environ deux fois moins de temps en subdivisant la surface plutôt que les volumes. Ainsi, si l'utilisateur n'a pas besoin de représenter la structure interne mais souhaite juste obtenir un fruit ou des volumes plus lisses (exemple : des graines), il choisira donc la subdivision surfacique pour de meilleures performances.

Cependant, bien que présentant de meilleures performances comparée à la subdivision volumique, la subdivision surfacique présente un inconvénient non négligeable. En effet, étant donné le fait que la subdivision de la surface d'un prisme entraîne la duplication de chacune de ses faces, le volume résultant ne peut plus être considéré comme un prisme et nous perdons alors toute information d'orientation. Ainsi, si nous prenons un prisme A dont la surface a été subdivisée par exemple, nous n'aurons alors aucune règle à notre disposition pour pouvoir ajouter un volume B sur une seule de ses faces E, puisque quatre

faces E seront ciblées par la règle $A \rightarrow A[B]_E$. Pour pallier cette limitation, nous pouvons soit définir un nouveau type de labels, extension qui sera présentée plus loin dans ce chapitre, ou soit utiliser la subdivision volumique. En effet, contrairement à la subdivision surfacique, en plus de pouvoir subdiviser la structure interne, elle présente l'avantage de conserver les informations d'orientation lorsqu'on l'applique. Les volumes résultants étant toujours des prismes, l'orientation de ces derniers est en effet conservée, et l'application de la règle $A \rightarrow A[B]_E$ par exemple, permettra toujours de cibler une seule face sur les volumes A qui sont créés.

2.4.2.3 Relations d'adjacence et nouveau paramètre de volume

En plus des deux règles de subdivisions surfaciques et volumiques, nous avons ajouté un attribut de subdivision S aux volumes de notre modèle. Ce paramètre remplit deux fonctions au sein d'un 3G-carte L-système. Premièrement, il permet de bloquer l'application de la subdivision sur certains volumes. Par exemple, si une règle doit subdiviser tous les volumes d'une 3G-carte, alors un volume $A(O, atx, aty, atz, H, L, W, S)$ ayant un paramètre S de valeur inférieure ou égale à zéro ($S_A \leq 0$) ne pourra pas être subdivisé. Au contraire, si la valeur de son paramètre de subdivision est strictement positive ($S_A > 0$), alors la subdivision pourra s'appliquer à ce volume.

Le deuxième rôle de l'attribut S consiste à influencer la subdivision lorsqu'elle est appliquée à des volumes adjacents. En effet, lorsque nous subdivisons une 3G-carte, il est important de conserver les relations d'adjacence, car c'est grâce à elles qu'il est possible de prendre en compte le contexte lors de la croissance. Dans notre système, lorsque deux volumes adjacents A et B sont subdivisés, le rétablissement de leur adjacence ainsi que leurs formes finales dépendent des valeurs de leurs attributs S_A et S_B. Ainsi, hormis le cas où A et B ne peuvent pas être subdivisés, nous pouvons identifier trois scénarios possibles :

Scénario 1 : $S_A > 0,\ S_B > 0\ et\ S_A \neq S_B$.
Dans ce cas de figure, A et B sont subdivisés. Cependant, en définissant une valeur différente de S pour chacun d'eux, nous voulons que ces deux volumes soient subdivisés comme si ils n'étaient pas adjacents. Pour ce faire, nous devons dans un premier temps les subdiviser séparément à l'aide du schéma de Catmull-Clark et sans prendre en compte le voisinage. Ensuite, nous recollons les faces censées être adjacentes en calculant les barycentres des sommets correspondants. La figure 2.27 illustre les étapes de ce processus.

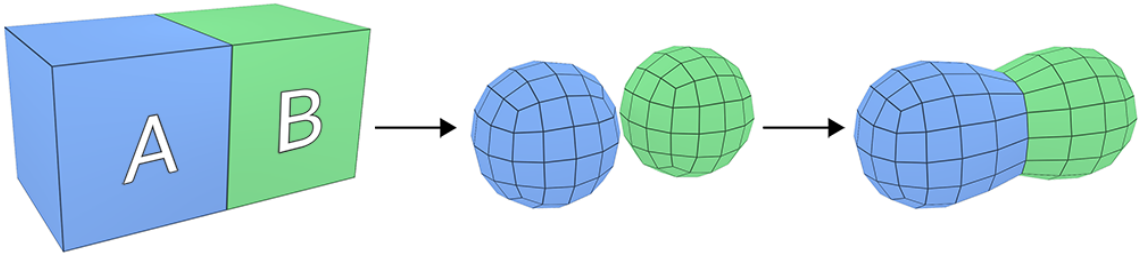


FIGURE 2.27 – Subdivision de volumes adjacents. Ici A et B sont subdivisés séparément.

Scénario 2 : $S_A > 0,\ S_B > 0\ et\ S_A = S_B$.
Ici, contrairement au premier scénario, en plus d'être subdivisés, les volumes A et B ont la même valeur de S. Cela signifie que nous voulons les subdiviser comme si ils ne formaient qu'un seul et même volume. Pour ce faire, A et B sont d'abord subdivisés sans déplacer les sommets de leur surface, puis les faces censées être adjacentes sont collées entre elles. Enfin, le schéma de Catmull-Clark est appliqué à la surface afin d'arrondir le modèle obtenu. Ainsi, le résultat de la subdivision est un modèle lisse, avec une jonction entre A et B qui n'est pas visible. Nous pouvons voir le résultat de ce procédé sur la figure 2.28.

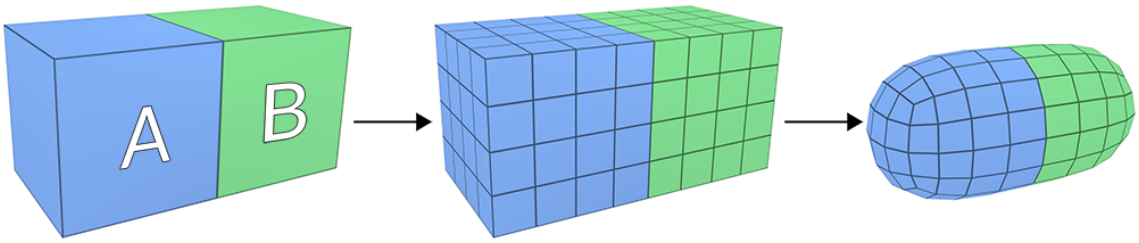


FIGURE 2.28 – Subdivision de volumes adjacents. Ici A et B sont subdivisés comme un seul et même volume.

Scénario 3 : $S_A > 0\ et\ S_B \leq 0\ (ou\ S_B > 0\ et\ S_A \leq 0)$.
Dans ce dernier scénario, A est subdivisé alors que B ne l'est pas (figure 2.29). Afin de conserver l'adjacence entre les deux volumes, puisque A est subdivisé, il nous faut subdiviser la face de B qui est incidente à A. Ainsi, une fois que la face est subdivisée, il nous suffit de coller les sommets du volume B à leurs sommets incidents sur le volume A.

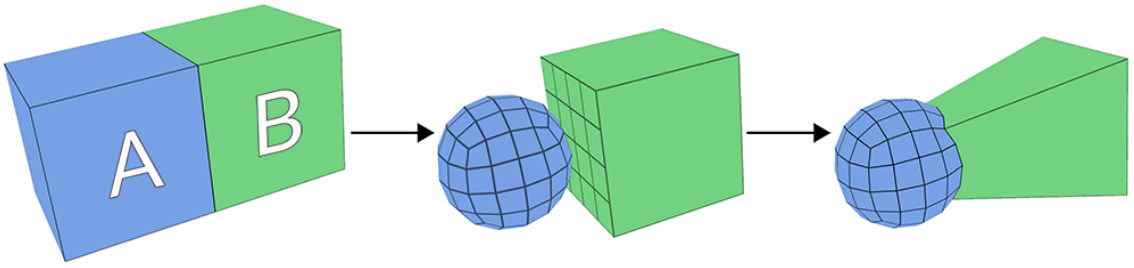


FIGURE 2.29 – Subdivision de volumes adjacents. Ici A est subdivisé, puis la face adjacente de B est subdivisé et collée au volume A.

2.4.2.4 Amélioration du contrôle de la croissance après la subdivision surfacique

Comme expliqué dans la section 2.4.2.2, la subdivision surfacique présente l'inconvénient majeur de dupliquer les faces des volumes. Par exemple, si nous subdivisons la surface d'un volume A d'ordre 4, ce dernier se retrouvera avec quatre faces E, quatre faces O et quatre exemplaires de chacune de ses faces latérales. En effet, cette caractéristique nous pose problème car, après la subdivision de A, il nous est alors impossible de pouvoir cibler une seule de ses faces avec des règles de production classiques. Pour corriger ce problème, nous avons ajouté un mécanisme de sous-labellisation à notre modèle.

Définition du concept

Dans notre modèle, un sous-label est un indice assigné aux faces lorsque la surface du volume auquel elles appartiennent est subdivisée. Cet indice consiste en une suite de caractères minuscules qui définissent la position d'une face au sein d'un ensemble de faces de même label. Ainsi, le mécanisme de sous-labellisation se déroule de la manière suivante. Lorsqu'un volume est créé, ses faces ont toutes un sous-label vide. Ensuite, à chaque fois que la surface de ce volume est subdivisée, une nouvelle lettre minuscule est ajoutée au sous-label de chaque face. La lettre à ajouter pour chaque face est définie en fonction de la position de cette dernière dans l'ensemble de faces qui partagent un label commun.

Par exemple, si nous subdivisons la surface d'un prisme d'ordre 4, sa face E sera remplacée par quatre faces labellisée E et sous-labellisées a, b, c et d : E_a, E_b, E_c et E_d (figure 2.30). De plus, si nous subdivisons la surface une seconde fois, alors la face Ea par exemple sera divisée en quatre faces : E_{aa}, E_{ab}, E_{ac} et E_{ad} (figure 2.30). Nous pouvons constater sur la figure 2.30 que la taille du sous-label d'une face permet de définir le nombre de subdivisions subies par le volume auquel elle appartient. Nous pouvons voir sur la figure 2.31 les sous-labels correspondant aux faces issues de trois subdivisions d'une face d'ordre 4.

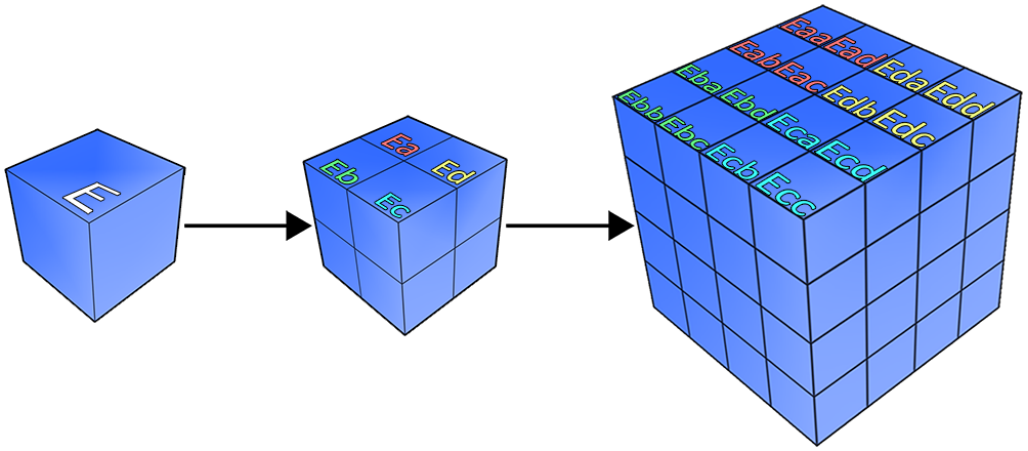


FIGURE 2.30 – Sous-labels correspondant à un prisme d'ordre 4 dont la surface est subdivisée une et deux fois.

Sélection de faces

Avec l'introduction de la sous-labellisation, nous avons pu étendre le mécanisme de sélection de faces grâce à l'utilisation d'expressions régulières [Fri06]. Par exemple, dans un système ne comprenant que des volumes A qui ont subi des nombres différents de subdivisions surfaciques, il est possible de sélectionner une seule face E ou un ensemble de faces E grâce à la formule $A_{E(F)}$. Ici, $E(F)$ représente toutes les faces E dont le sous-label correspond à une expression régulière représentée par F. Cette dernière respecte les normes du langage QRegExp du framework Qt [Com15], et peut prendre différentes formes en fonction des faces que nous voulons sélectionner. Voici quelques exemples de sélections de faces pour la formule $A_{E(F)}$:

- $E(\ \wedge[a-z]*\$\ )$: cette formule permet de sélectionner toutes les faces E, indépendamment de leurs sous-labels et du nombre de subdivisions subies par leur volume A. Ici, F peut avoir n'importe quelle taille, y compris zéro. Pour plus de simplicité, la formule A_E permet de sélectionner le même ensemble de faces que celle-ci.

- $E(\ \wedge[a-z]+\$\ )$: cette formule sélectionne le même ensemble de faces que la première formule, hormis les faces E des volumes n'ayant pas été subdivisés. En effet, dans cet exemple, à cause du caractère '+', l'expression régulière F doit être composée d'au moins un caractère.
- $E(\ \wedge\$\ )$: ici, F est strictement vide. Cela veut dire que nous ne sélectionnons que les faces E n'ayant pas de sous-label, c'est-à-dire celles qui appartiennent à des volumes qui n'ont pas été subdivisées.
- $E(\ bdca\ )$: dans cet exemple, nous ne sélectionnons qu'une seule face, c'est-à-dire celle qui est labellisée E et sous labellisée $bdca$.
- $E(\ \wedge b[a-z]*\$\ )$: ici, nous sélectionnons toutes les faces qui appartiennent à un volume ayant été subdivisé au moins une fois, et dont le sous-label commence par b. Pour un volume ayant été subdivisé trois fois par exemple, l'ensemble des faces sélectionnées par cette formule sont illustrées en rouge (foncé et clair) sur la figure 2.31.
- $E(\ \wedge[ab][ab][a-z]*\$\ )$: cette formule permet de sélectionner toutes les faces dont le sous-label commence par aa, ab, bb ou ba. Les volumes ciblés par cette formule auront donc été subdivisés au moins deux fois. Pour des faces issues de trois subdivisions d'une face d'ordre 4, celles qui seront sélectionnées par cette formule sont les faces des deux colonnes les plus à gauche sur la figure 2.31.
- $E(\ \wedge([ad]\{1,3\})\mid([dc]\{1,3\})\mid([cb]\{1,3\})\mid([ab]\{1,3\})\$\ )$: dans cet exemple, nous ne sélectionnons que des faces appartenant à des volumes subdivisés trois fois. De plus, les faces sélectionnées ont des sous-labels ne contenant soit que les lettres a et d, soit d et c, soit c et b ou soit a et b. En appliquant cette formule aux faces de la figure 2.31, l'ensemble des faces qui seront sélectionnées sont celles qui se trouvent sur les bords.

aaa	aad	ada	add	daa	dad	dda	ddd
aab	aac	adb	adc	dab	dac	ddb	ddc
aba	abd	aca	acd	dba	dbd	dca	dcd
abb	abc	acb	acc	dbb	dbc	dcb	dcc
baa	bad	bda	bdd	caa	cad	cda	cdd
bab	bac	bdb	bdc	cab	cac	cdb	cdc
bba	bbd	bca	bcd	cba	cbd	cca	ccd
bbb	bbc	bcb	bcc	cbb	cbc	ccb	ccc

FIGURE 2.31 – Ensemble de sous-labels obtenus après trois subdivisions d'une face d'ordre 4.

2.4.3 Opération de différenciation de cellules

Telle qu'elle est implémentée, la subdivision volumique ne nous permet pas à elle seule de générer les différents éléments de la structure interne d'un fruit. Sur la figure 2.32 par exemple, nous pouvons voir ce qui se passe au sein d'une sphère lorsque son volume est subdivisé deux fois.

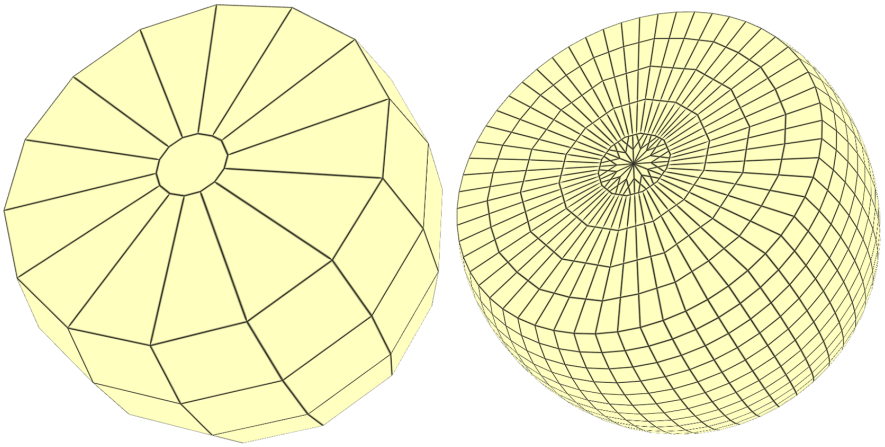


FIGURE 2.32 – Aperçu de la structure interne d'une sphere subdivisée deux fois.

Après application de la subdivision volumique, la structure interne du fruit est bien composée de plusieurs volumes (ou cellules), mais rien ne nous permet de définir la couche que chacun de ces derniers est censé représenter. De plus, la structure interne ne contient aucune cellule de forme arrondie censée représenter les graines. Pour résoudre ces problèmes, nous avons introduit une nouvelle opération appelée : opération de différenciation. Tout comme la phase de différenciation qui a lieu au cours de la croissance d'un fruit (voir chapitre 1 - section 1.1.2), cette opération va permettre à toutes les cellules de la structure arborescente de se différencier les unes des autres, de manière à ce que chacune d'entre elles fasse partie d'un élément bien précis de la structure interne du fruit. Pour appliquer cette opération nous utilisons la règle de production suivante :

$$B \rightarrow (C, 12, D, 15, B, 50, 7, G) \tag{2.2}$$

L'application de la règle 2.2 va permettre de différencier tous les volumes B de la structure arborescente pour former les couches du péricarpe du fruit ainsi que les graines de ce dernier. Pour ce faire, la règle prend huit paramètres en entrée : le label à attribuer aux volumes représentant l'exocarpe, l'épaisseur de ce dernier, le label à attribuer aux volumes qui représentent le mésocarpe, l'épaisseur du mésocarpe, le nom à donner aux volumes qui forment l'endocarpe, l'épaisseur de cette couche, le nombre de carpelles qui forment le fruit et enfin le label à attribuer aux volumes qui forment les graines. Pour mieux comprendre le fonctionnement de cette opération, voici ce qui se passe lorsque nous appliquons la règle 2.2 au fruit de la figure 2.32 :

1. Dans un premier temps, tous les volumes B qui se trouvent sur les bords du modèle 3D sont renommés en volumes C afin de les différencier des autres et former ainsi l'exocarpe. Sur la figure 2.33, ce changement de label se matérialise par un changement de couleur au niveau des volumes du bord. Ensuite, une fois que le changement de label a été effectué, la taille des volumes C fraîchement créés s'adapte de manière à ce que la couche qu'ils forment ait la taille spécifiée par les paramètres de la règle, soit 12 unités de longueur dans notre exemple. Pour ce faire, on récupère les faces qui se trouvent de l'autre côté des volumes C par rapport aux faces du bord, et on les déplace le long des arêtes qui leurs sont incidentes. Par exemple, si pour un volume C donné, sa face se trouvant au bord est une face E, alors la face à déplacer et qui est de l'autre côté de C par rapport à E est la face O.

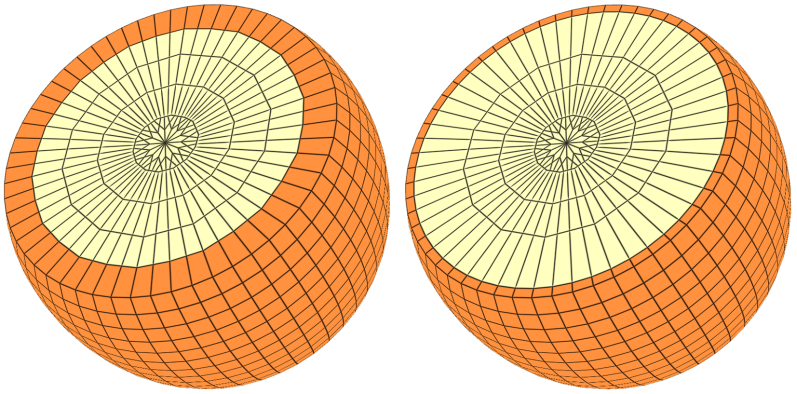


FIGURE 2.33 – Première étape de l'application de l'opération de différenciation : création de l'exocarpe.

2. Une fois que l'exocarpe est formé, tous les volumes B qui sont directement adjacents à cette couche se différencient et changent ainsi de label pour devenir des volumes D, afin de pouvoir représenter le mésocarpe. Comme pour la première étape, s'en suit ensuite un déplacement des faces qui se trouvent de l'autre côté des faces adjacentes à l'exocarpe, jusqu'à ce que leurs arêtes mesures 15 unités de longueur, comme il a été spécifié dans la règle 2.2.

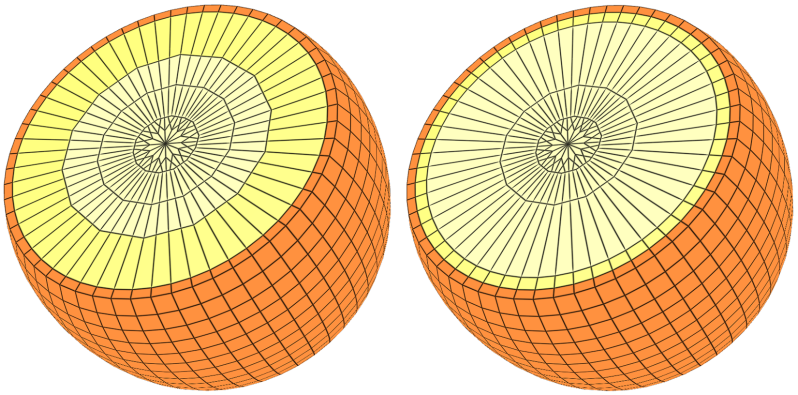


FIGURE 2.34 – Seconde étape de l'application de l'opération de différenciation : création du mésocarpe.

3. Lorsque le mésocarpe a été formé, seule une partie des volumes B qui sont adjacents à cette couche vont se différencier, contrairement aux deux premières étapes. Dans notre exemple, nous voulons que le fruit final comporte sept loges. Pour les représenter correctement, le système va compter le nombre de volumes B issus du tronc et va le diviser afin de déterminer le nombre de volumes à utiliser pour former chaque loge. Ici, cinquante-six branches sont collées à chaque étage du tronc. Ainsi, en divisant ce nombre par sept, on en déduit que ces branches seront divisées en sept loges comportant chacune huit branches. Aussi, étant donné qu'au sein d'un fruit les loges sont séparées les unes des autres par une paroi formée par le mésocarpe, plutôt que de différencier tous les volumes B restant, le système ne différenciera que ceux qui se trouvent au niveau des parois des loges. Pour chaque loge de huit branches, on obtient alors six branches de label B qui représentent l'endocarpe, et deux branches de label D qui représentent les parois formées par le mésocarpe.

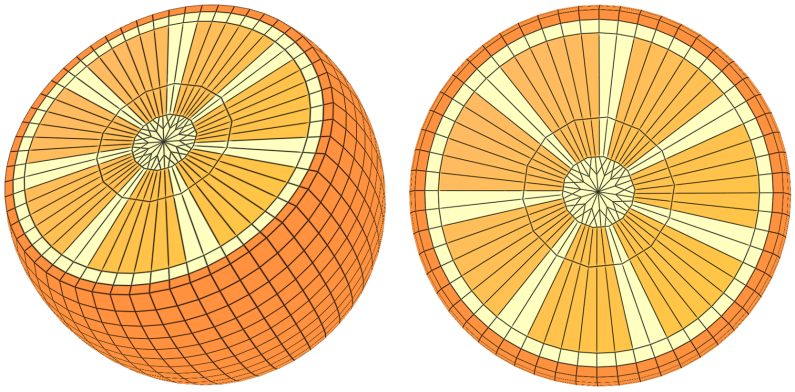


FIGURE 2.35 – Troisième étape de l'application de l'opération de différenciation : création de l'endocarpe et des loges.

4. Finalement, la dernière étape consiste à créer les graines au sein des loges. Pour ce faire, le système va choisir aléatoirement un ensemble de volumes B qui sont collés au tronc, va les différencier en volumes G et va y appliquer une subdivision surfacique pour qu'ils puissent représenter les graines.

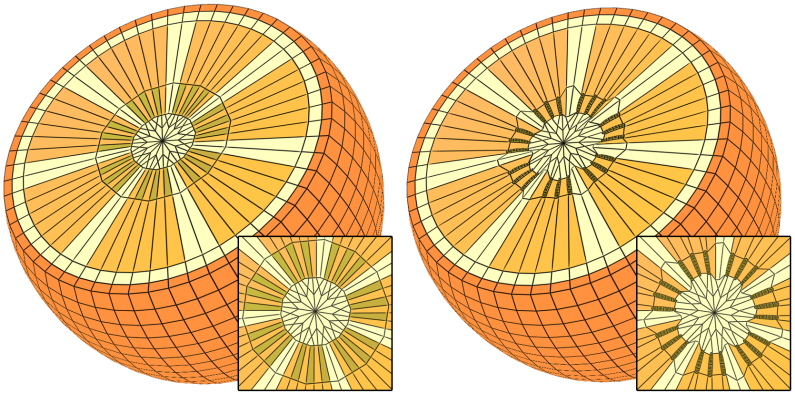


FIGURE 2.36 – Quatrième et dernière étape de l'application de l'opération de différenciation : création des graines.

2.4.4 Synthèse et Résultats

Dans cette section, nous avons introduit une nouvelle extension au concept de 3G-cartes L-systèmes qui est basée sur la subdivision de Catmull-Clark. Ce concept, couplé à notre modèle nous a ainsi permis d'obtenir un outil puissant capable d'augmenter la précision du processus de modélisation, tout en maintenant le côté intuitif des 3G-cartes L-systèmes. L'ajout de cette extension nous a ainsi apporté la possibilité de répondre à certaines problématiques de la modélisation de fruits.

Premièrement, la subdivision surfacique nous permet d'obtenir des formes de fruits qui sont plus lisses et plus arrondies de manière relativement simple. En effet, comme nous pouvons le voir sur la figure 2.38, plutôt que de générer directement la forme d'un fruit, il est possible de créer dans un premier temps une forme grossière qui sera ensuite affinée grâce à l'utilisation de la subdivision surfacique.

Dans un second temps, nous avons introduit l'utilisation d'un paramètre S dans la définition des volumes, permettant d'influer sur le comportement de ces derniers face à la subdivision. Grâce à ce nouveau paramètre, nous avons maintenant la possibilité de générer des graines au sein de nos fruits. Comme nous pouvons le voir sur la figure 2.39 par exemple, seuls les volumes censés représenter des graines sont affectés par la subdivision surfacique, ce qui permet de leur donner une forme plus lisse que les autres volumes.

Le mécanisme de sous-labellisation représente la troisième contribution apportée au modèle. Ce mécanisme permet de garder le contrôle sur le processus de modélisation grâce à l'utilisation d'expressions régulières pour sélectionner des ensembles de faces. Cette extension nous a par exemple été utile pour modéliser une cerise ainsi que sa structure interne (figure 2.40). Pour ce faire, nous avons subdivisé l'axiome de départ afin d'obtenir un noyau, sur lequel nous avons ajouté plusieurs volumes dont les tailles variaient en fonction de leurs sous-labels.

Enfin, la dernière contribution apportée par l'ajout de la subdivision est l'opération de différenciation. Grâce à cette dernière, l'ensemble des trois contributions précédentes ont pu être combinées au sein d'une même règle afin de générer la structure interne des fruits créés à l'aide de notre système. Nous pouvons voir sur la figure 2.37 le rendu d'un modèle complet de tomate à deux loges réalisé à l'aide du logiciel *Autodesk 3D Studio Max 2013*.

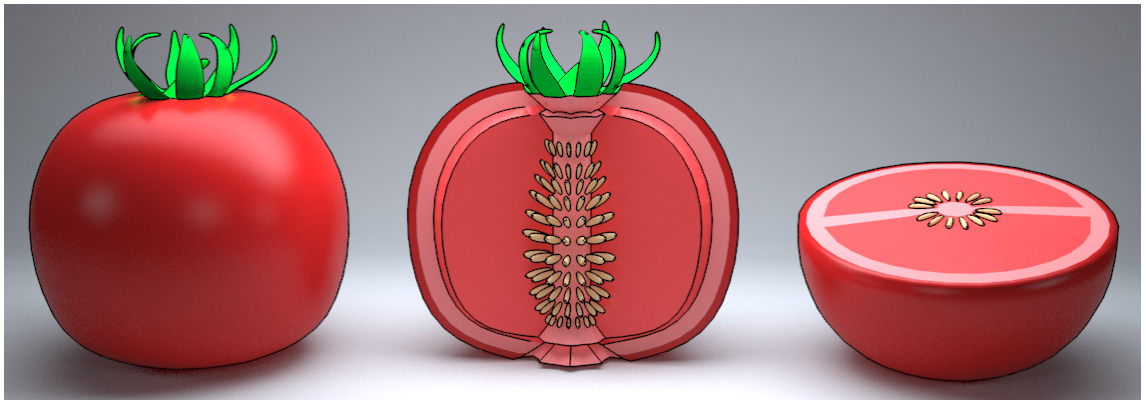


FIGURE 2.37 – Rendu *Autodesk 3D Studio Max 2013* des coupes d'une tomate ronde créée à l'aide de notre système.

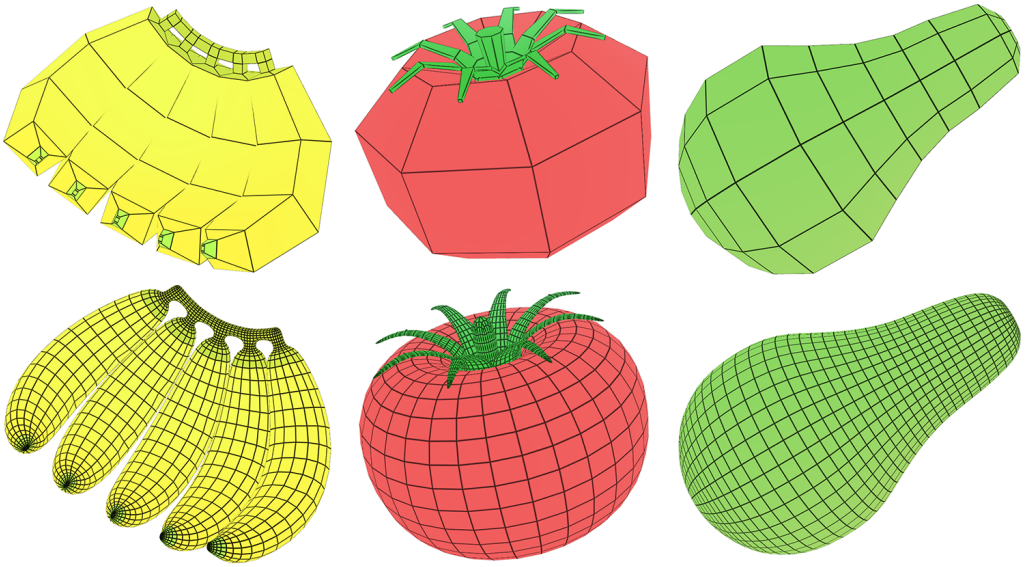


FIGURE 2.38 – Des fruits dont la forme est lissée grâce à la subdivision surfacique.

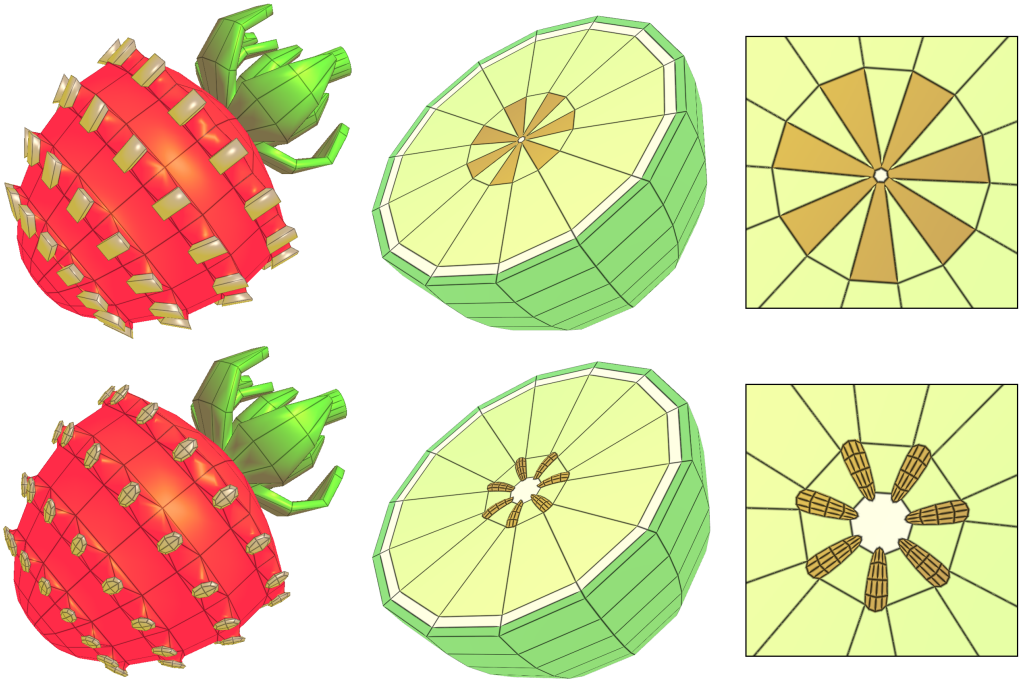


FIGURE 2.39 – Création de graines à l'aide du paramètre de subdivision des volumes.

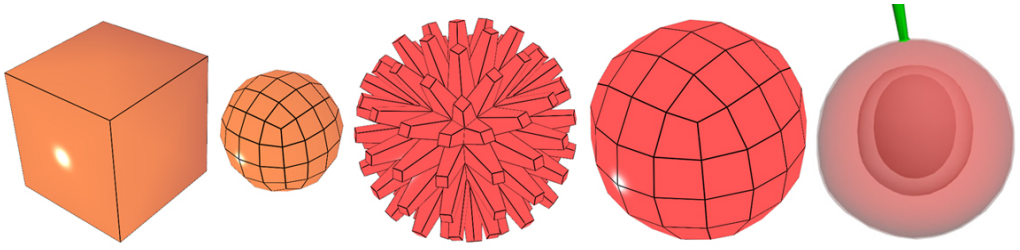


FIGURE 2.40 – génération d'une cerise et de sa structure interne à l'aide de la sous-labellisation.

2.5 Conclusion

À travers ce chapitre, nous avons montré que les 3G-cartes L-systèmes se prêtaient bien à la modélisation de fruits simples charnus et de leur structure interne. Dans un premier temps nous avons introduit un 3G-carte L-système basique permettant de modéliser ce type de fruit. En se basant sur l'observation qu'une grande majorité d'espèces de fruits présente une symétrie rotationnelle autour d'un axe, nous avons proposé un système faisant croître un tronc central autour duquel des branches sont générées puis collées entre elles pour former le péricarpe du fruit. Bien que donnant des résultats pertinents, la modification de la grammaire de ce système s'est révélée être une tâche beaucoup trop fastidieuse lorsque nous voulions modéliser une autre espèce de fruit. Ainsi, pour pallier ce problème, nous avons donc introduit une paramétrisation de notre système. Pour ce faire, nous avons démontré qu'il était possible de décomposer la forme relativement complexe d'un fruit en plusieurs formes simples, représentées par des fonctions mathématiques basiques. La combinaison de ces fonctions nous a ainsi permis de créer un 3G-carte L-système facilement paramétrable et permettant de créer des variations globales du résultat final de manière relativement simple.

Dans la seconde partie de ce chapitre, nous avons introduit une méthode permettant de représenter la structure interne de nos fruits de manière simple et intuitive. Pour ce faire, nous avions trois objectifs. Premièrement, il nous fallait être capables de représenter les trois couches du péricarpe : l'exocarpe, le mésocarpe et l'endocarpe. Deuxièmement, nous voulions pouvoir générer des graines au sein de nos modèles 3D. Et enfin, nous voulions que la génération de ces éléments de la structure interne se fasse sans influencer la surface du fruit déjà mise en place par notre système paramétrique. Pour répondre à toutes ces problématiques, nous avons introduit l'utilisation de la subdivision de Catmull-Clark au sein des 3G-cartes L-systèmes. L'ajout de ce concept à notre modèle nous a permis d'introduire plusieurs extensions. La première, la subdivision surfacique, nous a donné la possibilité d'obtenir des surfaces lisses de nos fruits sans avoir à utiliser un très grand nombre de volumes. De plus, elle nous a aussi permis d'arrondir la forme des volumes de nos systèmes de manière à pouvoir former des graines. La seconde extension, la subdivision volumique, nous a quant à elle permis de diviser la structure arborescente de notre système en plusieurs petites cellules, nous donnant ainsi la possibilité de modifier ces dernières de manière à représenter tous les éléments de la structure interne. Enfin, nous avons ajouté une règle de différenciation à notre système permettant de générer tous les éléments de la structure interne de manière automatique et ce, sans modifier la forme de la surface déjà mise en place par notre système paramétrique.

Chapitre 3

Génération de défauts géométriques de fruits : le cas de la tomate

Sommaire

Chapitre 3

Génération de défauts géométriques de fruits : le cas de la tomate

3.1 Introduction

Le réalisme fait très certainement partie des principaux objectifs de la recherche en informatique graphique. Pour l'atteindre, les méthodes existantes tentent d'apporter des solutions à des problématiques qui sont diverses et variées. Parmi elles, la simulation du vieillissement et la création d'imperfections sur les surfaces jouent un rôle important. En effet, sauf lorsque nous cherchons à créer des objets propres et exempts de tous défauts, l'absence de ces derniers peut mener à une apparence non réaliste due à une surface qui est trop propre et trop lisse.

Dans le cas des fruits, ces imperfections sont appelées défauts de fruits, que nous avons précédemment définis comme étant des caractéristiques qui affectent leur aspect, leur comestibilité ou leur valeur commerciale. Nous pouvons classer ces défauts en deux catégories : les défauts d'apparence, généralement représentés par une variation de couleur ou par la présence de corps étrangers au sein du fruit (bactéries, champignons, etc...), et les défauts géométriques, généralement caractérisés par des variations au niveau de la forme du fruit. En informatique graphique, la première catégorie sera généralement représentée par des méthodes de rendu, comme c'est le cas notamment avec la méthode de Kider *et al.* [KRB11], alors que les défauts géométriques seront quant à eux souvent représentés par des méthodes de modélisation géométrique, comme dans [LCW+12].

Dans cette partie de nos travaux, nous nous sommes focalisés sur la représentation des défauts géométriques spécifiques à la tomate. Ainsi, en se basant sur les normes définies dans [UNE12] et dans [OEC02], nous avions plusieurs objectifs. Premièrement, la reproduction de tomates ayant des formes dégénérées. Deuxièmement, la création de variations géométriques locales, appelées excroissances, à des endroits spécifiques de la surface d'une tomate. Enfin, la génération de défauts géométriques tels que les crevasses ou les impacts de grêle. Pour ce faire, nous avons introduit plusieurs extensions au concept de 3G-cartes L-systèmes dans le but de pouvoir générer des défauts sur nos fruits obtenus à l'aide de notre système.

3.2 Problématique

Dans le chapitre précédent, nous avons proposé un modèle permettant de généraliser la forme des fruits simples charnus afin de pouvoir générer une grande variété de fruits appartenant à plusieurs espèces différentes. De plus grâce à notre 3G-carte L-système paramétrique, nous avons introduit la possibilité de décrire la forme générale d'un fruit comme étant une combinaison de variables et de fonctions mathématiques définies dans nos grammaires. Cette extension nous a ainsi permis de faire varier la forme des modèles générés par notre système de manière relativement simple et intuitive. Cependant, bien que notre modèle soit pertinent pour générer plusieurs fruits appartenant à des espèces différentes, la forme générale de ces derniers est souvent trop régulière et parait trop parfaite.

Afin d'obtenir des résultats à l'apparence plus réaliste, nous voulons être capables de représenter des imperfections géométriques sur les tomates qui ont été générées par notre 3G-carte L-système. Ainsi, dans un premier temps, nous aimerions pouvoir perturber la forme générale du fruit sans avoir à modifier les fonctions qui permettent de générer sa forme standard. Comme nous pouvons le voir sur la figure 3.1, une telle approche nous permettrait alors de dissocier l'étape de génération de la forme standard du fruit, des deux étapes qui permettent de générer des défauts sur ce dernier. Pour ce faire, nous avons introduit dans notre modèle des nouvelles fonctions qui vont permettre de faire varier la forme du résultat final de manière aléatoire. De plus, afin de garder un certain contrôle sur le processus de modélisation, nous introduirons également un mécanisme permettant de contraindre la forme du fruit à respecter certaines proportions définies par l'utilisateur.

En plus d'une variation plus poussée de la forme générale du fruit, nous voulons pouvoir cibler des parties spécifiques de sa surface de manière à pouvoir y créer des variations géométriques locales. Chez la tomate en particulier, il en existe plusieurs types. Dans les travaux présentés dans ce chapitre, nous avons introduit des extensions au concept de 3G-cartes L-systèmes permettant de créer quatre types de défauts spécifiques. Ainsi, nous allons présenter dans un premier temps une méthode permettant de cibler des parties spécifiques de la surface d'une tomate dans le but d'y générer des excroissances. Ensuite, nous introduirons une nouvelle règle de production qui nous permettra de déchirer la surface du fruit dans le but de créer des crevasses. Puis, nous présenterons une autre règle de production permettant de générer des impacts de grêle aléatoirement sur la surface du fruit. Et nous finirons enfin par présenter un moyen de simuler l'affaissement du fruit.

La figure 3.1 montre la version étendue de notre pipeline. L'utilisateur fournit une grammaire dans laquelle sont définies la forme du fruit ainsi que ses imperfections géométriques. Notre modèle crée alors dans un premier temps sa forme standard, qu'il fait varier à l'aide des paramètres de défauts globaux. Puis, il génère la structure interne du modèle 3D. Enfin, il ajoute des variations géométriques locales à la surface du fruit.

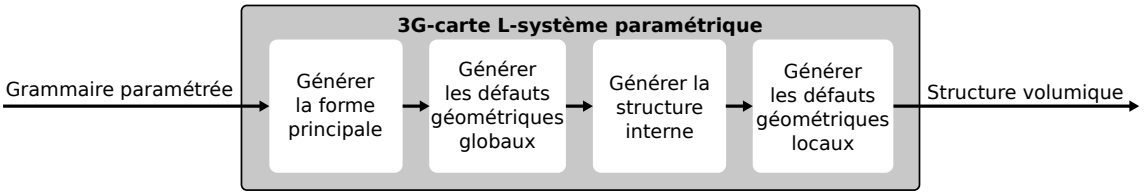


FIGURE 3.1 – Ajout de la génération de défauts aux étapes de l'exécution de notre modèle.

3.3 Génération de tomates mal formées

Le fruit peut être vu comme le résultat de l'enchaînement d'une multitude de processus physiologiques complexes qui interagissent fortement entre eux (respiration, transpiration, expansion cellulaire, etc...). Ainsi, lorsque des facteurs viennent entraver le bon déroulement de l'un de ces processus (exemple : un climat défavorable ou une insuffisance d'eau), cette perturbation engendre généralement un dérèglement du processus de croissance, ce qui a souvent pour conséquence l'apparition d'anomalies (ou défauts) sur le fruit mature. Dans cette section nous présenterons deux méthodes différentes permettant de représenter un type d'anomalies spécifique : les malformations de la forme générale d'une tomate. La première méthode nous permettra de générer des malformations de formes aléatoires, alors que la seconde privilégiera le contrôle de la forme de ces anomalies.

3.3.1 Première approche : intégration de l'aléatoire dans notre système

La difficulté lorsque nous voulons générer des malformations sur nos fruits réside dans le fait que, de par l'équilibre fragile du processus de croissance, le nombre de formes possibles est très élevé, voire illimité, leur donnant ainsi un caractère aléatoire. De plus, bien que nous savons que ces malformations sont généralement dues soit à un climat défavorable ou soit à un usage excessif de pesticides (voir [Bla09]), la prédiction de l'apparence d'une malformation est un domaine qui est encore très peu documenté, augmentant ainsi la difficulté de nos objectifs à atteindre. La figure 3.2 illustre quelques exemples de malformations qu'il est possible d'observer chez différentes variétés de tomates.

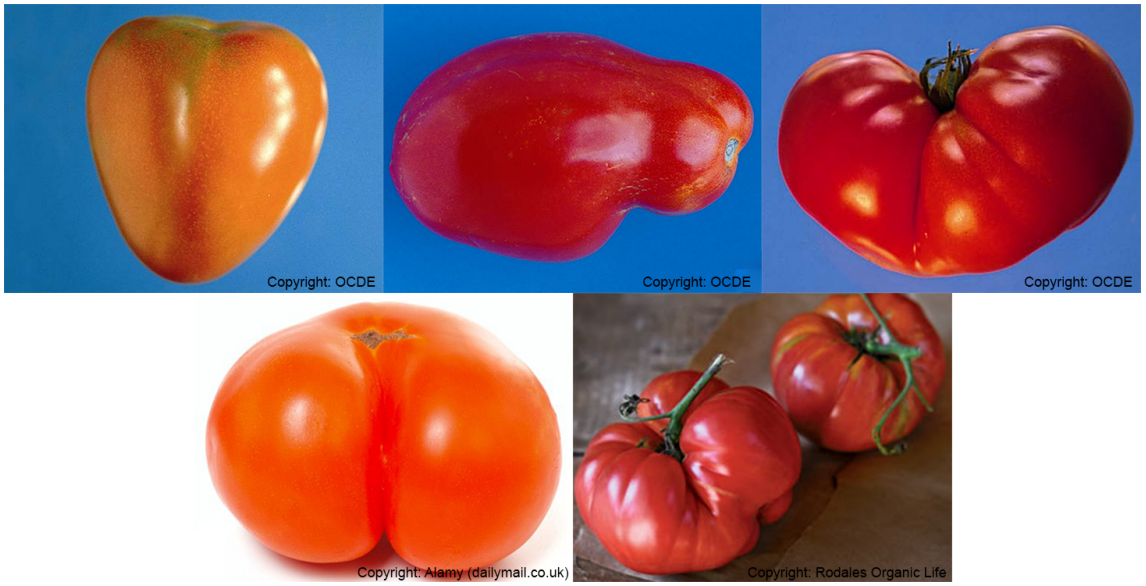


FIGURE 3.2 – Tomates mal formées. De gauche à droite et de haut en bas : une tomate ronde, une tomate allongée, une tomate côtelée, une autre tomate ronde et d'autres tomates côtelées.

Afin de pouvoir créer des variations au niveau de la forme d'une tomate, il nous faut intervenir sur la phase de génération de la structure arborescente de notre système paramétrique. Pour coller au mieux au caractère aléatoire des malformations, une première approche, relativement simple à mettre en place, consiste à ajouter de l'aléatoire au niveau de la définition des paramètres de notre système. Cette méthode nous permettra ainsi d'obtenir un fruit de forme différente à chaque exécution. Et lorsque les valeurs des différents paramètres ne seront pas en harmonie les unes avec les autres, cela se répercutera alors sur la forme finale, qui pourra être comparée avec les normes établies dans [OEC02] afin de définir si la tomate peut être considérée comme étant mal formée ou pas. Dans la grammaire 3.1, nous montrons comment ajouter de l'aléatoire au niveau de la définition des variables de notre système, et la figure 3.3 illustre une vingtaine de résultats obtenus qui, selon les normes de [OEC02], peuvent être définis comme des tomates présentant une malformation.

Grammaire 3.1 : `Ajout de l'aléatoire aux valeurs des variables.`

@Définition des volumes@
$\#define\ A(10, 1, 0, 0, 0, 1, 5, 5)$
$\#define\ B(4, 1, 0, 0, 0, 70, 2, 5)$

@Paramétrisation de l'aléatoire@
$\#function\ variation(x, taux) \quad return\ x + random(-x \times taux \ , \ x \times taux);$
$\#function\ randomval(x, y) \quad return\ random(x \ , \ y);$

@variables générales@
$\#define\ precision = variation(10\ ,\ 50\%)$
$\#define\ ph\ =\ variation(11\ ,\ 50\%)$
$\#define\ largeur = variation(55\ ,\ 20\%)$
$\#define\ longueur = variation(15\ ,\ 10\%)$
$\#define\ creux\ =\ variation(0.5\ ,\ 10\%)$
$\#define\ nbprotus\ =\ randomval(1\ ,\ 8)$
$\#define\ anglesup = randomval(0\ ,\ 20)$
$\#define\ angleinf = randomval(-60\ ,\ -30)$
$\#define\ nbvolangle = variation(5\ ,\ precision)$

@Définition des fonctions, de l'axiome et des règles@
...

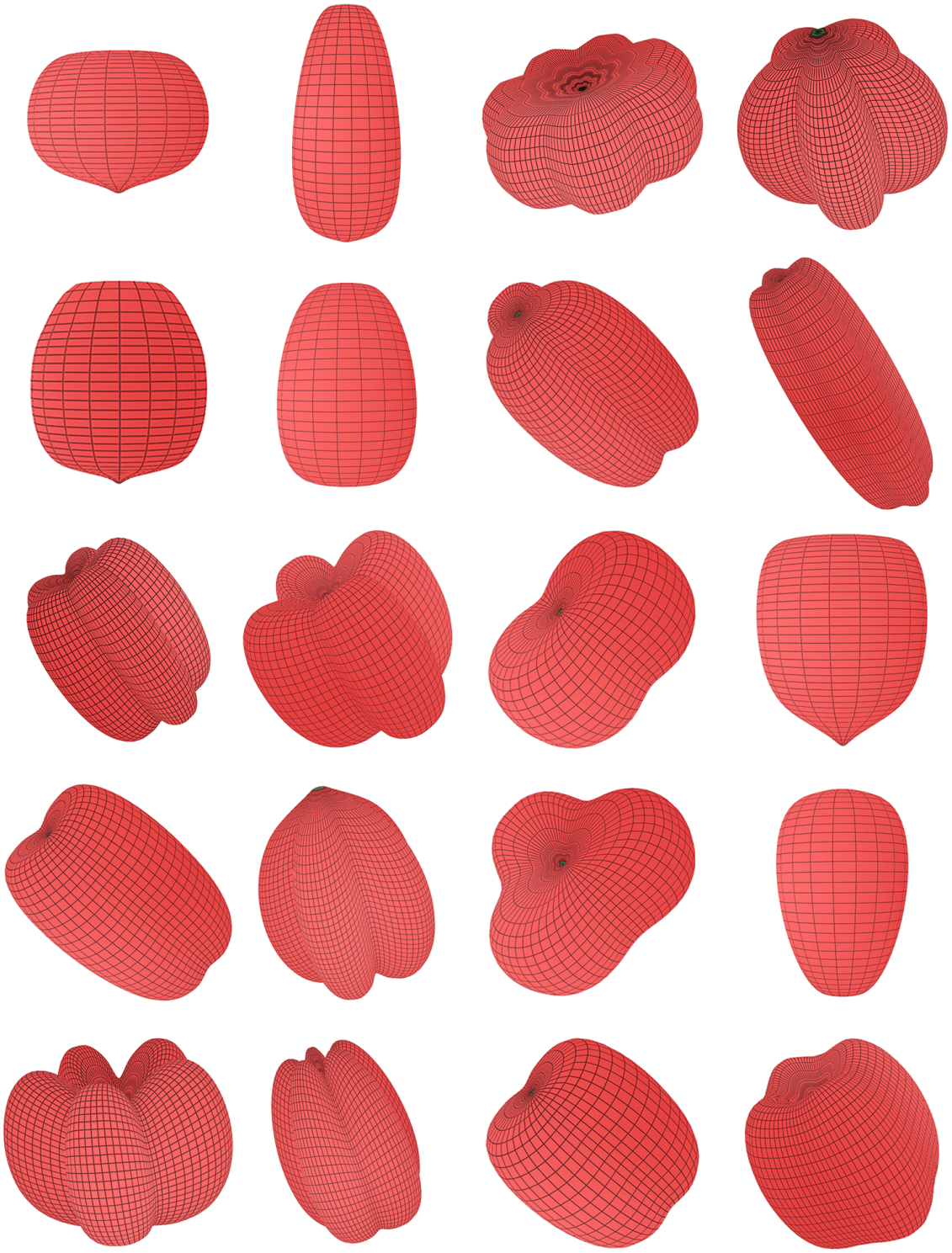


FIGURE 3.3 – Différents résultats obtenus après exécution de la grammaire 3.1. Ici, grâce à l'aléatoire, la forme du résultat final change à chaque exécution du système.

Le système représenté par la grammaire 3.1 est une version étendue de notre système paramétrique présenté dans le chapitre précédent (voir grammaire 2.7). Ainsi, afin de faciliter la lecture et la compréhension de cette version étendue, les parties qui sont communes aux deux grammaires ne sont pas représentées ici (fonctions, règles de production, etc...).

Tout comme le système du chapitre précédent, celui-ci génère donc un tronc composé de volumes A ajoutés les uns sur les autres, et autour desquels des branches B vont être créées. De plus, à chaque itération du système, différentes fonctions et variables mathématiques vont venir influencer la génération de cette structure arborescente en attribuant des valeurs spécifiques aux paramètres de chaque volume créé. Cependant, la différence avec la version précédente réside dans l'introduction de deux nouvelles fonctions qui permettent de faire varier de manière aléatoire les valeurs des variables du système.

La première fonction que nous avons introduit, appelée $variation(x, taux)$, permet à l'utilisateur de définir pour une variable donnée une valeur par défaut ainsi qu'un taux de variation exprimé en pourcentage. Dans la grammaire 3.1 par exemple, nous avons défini la valeur par défaut de la variable $precision$ à 10, avec un taux de variation de 50%. Ainsi, à chaque nouvelle exécution de notre système, la variable $precision$ se verra attribuer une valeur aléatoire comprise entre 5 et 15. Quant à la deuxième fonction introduite dans notre système, appelée $randomval(x, y)$, elle permet de faire un simple appel à $random(x, y)$, une fonction du langage C qui génère de manière pseudo-aléatoire un nombre compris entre x et y. Ainsi, à chaque exécution du système représenté par la grammaire 3.1 par exemple, le nombre de protubérances du fruit final sera compris entre 1 et 8.

Comme nous pouvons le voir sur la figure 3.3, l'introduction de ces deux nouvelles fonctions présente l'avantage de pouvoir générer une multitude de formes de tomates différentes, sans avoir à modifier les variables de la grammaire avant chaque exécution du système. De plus, grâce au fait que nous ne modifions pas les fonctions mathématiques définies dans le chapitre précédent, l'utilisateur peut toujours obtenir la forme standard du fruit sans avoir à effectuer beaucoup de changements au sein de la grammaire. Ainsi, il lui suffira par exemple de définir un taux de variation égal à 0% pour annuler les effets de la fonction $variation(x, taux)$, alors qu'il n'aura qu'à donner une valeur similaire aux paramètres x et y de la fonction $randomval(x, y)$ pour qu'elle n'influence plus les valeurs des variables du système.

Cependant, cette approche présente une limitation majeure. En effet, bien qu'elle nous permette de générer une forme différente à chaque nouvelle exécution du système, l'utilisation de fonctions aléatoires limite grandement le contrôle et la prédictibilité du résultat final. Ainsi, comme nous pouvons le voir sur la figure 3.3, les vingt formes obtenues à l'aide de notre système sont toutes très différentes et, si l'utilisateur souhaite obtenir une forme de malformation spécifique, il lui faudra très certainement effectuer plusieurs exécutions du système jusqu'à obtenir la forme souhaitée. Pour pallier cette limitation, nous avons introduit une seconde approche qui permet d'augmenter le contrôle du processus de modélisation et la prédictibilité du résultat final.

3.3.2 Deuxième approche : ajout de contraintes au résultat final

Les fonctions $variation(x, taux)$ et $randomval(x, y)$ nous permettent de faire varier la forme du résultat final d'une exécution à l'autre de manière relativement simple. Cependant, bien qu'il soit possible de contrôler assez facilement l'ensemble des valeurs pouvant être attribuées à une variable spécifique, l'ajout d'une composante aléatoire à toutes les variables du système peut rapidement augmenter la difficulté du contrôle sur la forme du résultat final. Pour palier ce problème nous proposons d'utiliser des fonctions qui vont permettre de contraindre la forme du fruit. Ces fonctions, appelées fonctions contraintes, sont inspirées de la méthode de Huang *et al.* qui, pour contrôler la forme d'une grappe de raisin, proposent dans [HJT+13] d'utiliser un octaèdre dont les faces délimitent les zones que les baies ne doivent pas dépasser lors de leur croissance. Dans notre modèle, plutôt que d'utiliser un polyèdre pour définir la forme du fruit, nous allons utiliser deux fonctions contraintes, l'une définie le long du tronc et l'autre définie autour du tronc qui, une fois combinées entre elles, permettront de définir une surface, appelée surface englobante, et qui contiendra la forme finale du fruit.

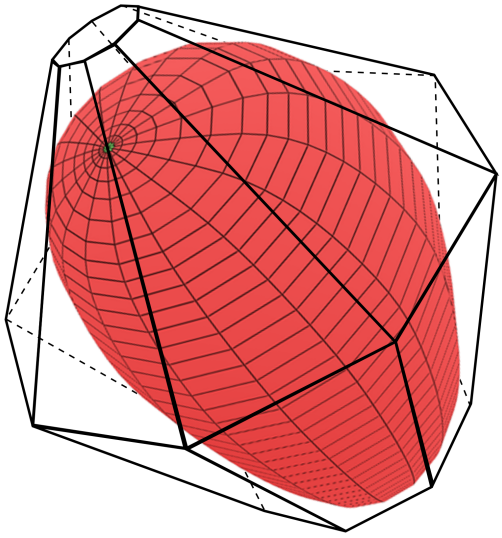


FIGURE 3.4 – Représentation de la surface qui contraint la forme du fruit. Cette surface est représentée par deux prismes d'ordre égal à celui du tronc et qui sont ajoutés l'un sur l'autre.

3.3.2.1 Définition de la fonction contrainte verticale

La première fonction contrainte à définir est appelée fonction contrainte verticale. Lorsque nous avons présenté la paramétrisation du système dans le chapitre précédent, nous avons montré comment paramétrer la taille des branches en fonction de l'étage du tronc à partir duquel elles sont générées. Ici, la fonction contrainte verticale joue un rôle similaire. En effet, plutôt que définir une taille pour chaque branche lors de sa génération, celle-ci va la contraindre à ne pas sortir de la surface englobante et à respecter les proportions définies par cette dernière. Ainsi, lorsque la taille d'une branche ne respectera pas les contraintes mises en place par la surface englobante, sa taille par défaut sera alors modifiée par les fonctions contraintes. Nous montrons dans la grammaire 3.2 comment intégrer la fonction contrainte verticale à notre système.

Grammaire 3.2 : `Ajout de la fonction contrainte verticale.`

@Définition des volumes, fonctions et variables@

...

@Fonction contrainte verticale@

$\#define\ stepmax = 13$

$\#define\ vmax = 1$

$\#define\ vmin_a = 0.1$

$\#define\ dx_a = (vmax - vmin_a)/stepmax$

$\#function\ variate_a(x)\ return\ vmin_a + dx_a \times x;$

$\#define\ vmin_b = 0.8$

$\#define\ dx_b = (vmax - vmin - b)/(precision - stepmax)$

$\#function\ variate_b(x)\ return\ vmax - dx_b \times (x - stepmax);$

$\#function\ variate(x)\ return\ x \leq stepmax\ ?\ variate_a(x)\ :\ variate_b(x);$

@Définition de l'axiome et des règles de production@

...

$p03\ A\{\}\{etape < precision\}\{\} \rightarrow A[B(,,,,,< taille(etape) \times variate(etape) >,,)]_{C*}$

...

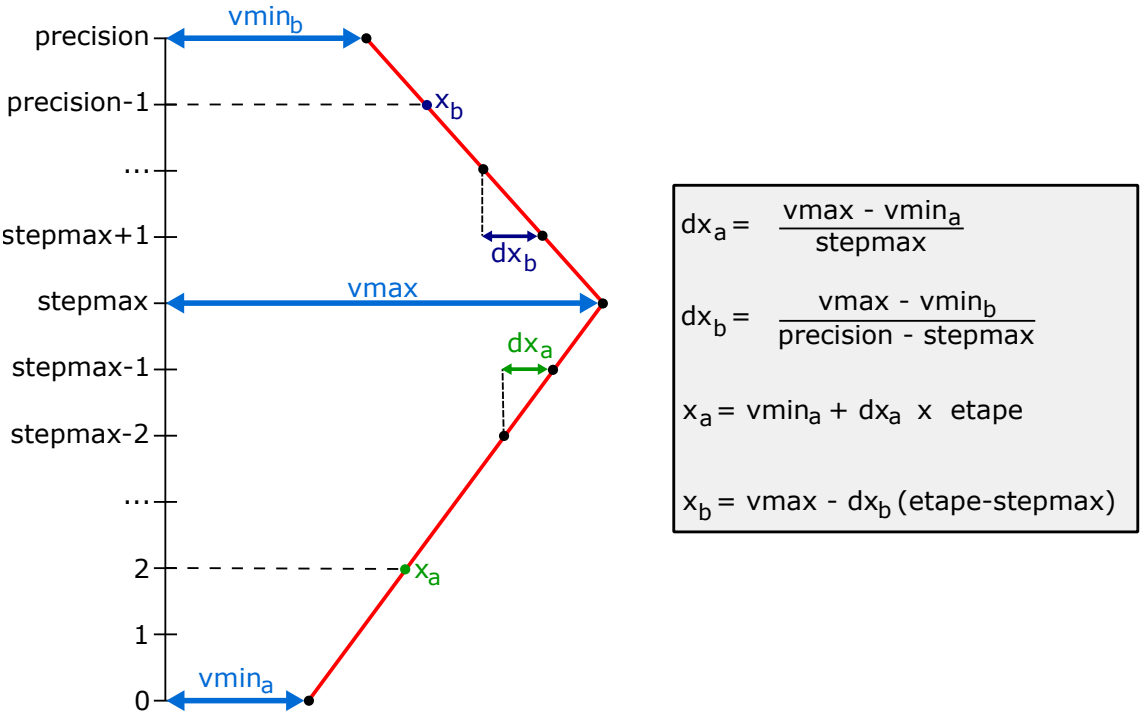


FIGURE 3.5 – Évolution du coefficient de variation x en fonction de l'étage auquel nous nous trouvons dans le tronc.

Nous introduisons dans la nouvelle version de notre système, représentée par la grammaire 3.2, l'utilisation d'une fonction contrainte verticale appelée $variate(x)$. À chaque itération du système, cette fonction renvoie un facteur de variation dont la valeur dépend du nombre d'itérations déjà subies par le système. La valeur ainsi obtenue est alors multipliée par la taille de toutes les branches générées lors de cette étape, afin de coller aux proportions définies par la surface englobante.

Dans notre système, la fonction $variate(x)$ est paramétrée par quatre variables. Les variables $vmin_a$ et $vmin_b$ permettent de définir les valeurs des facteurs à appliquer respectivement aux branches du premier et du dernier étage du tronc. La variable $vmax$ permet quant à elle de définir la valeur maximale du facteur de variation, qui sera appliquée aux branches générées sur le volume A situé à l'étage $stepmax$ du tronc. Ainsi, comme nous pouvons le voir sur la figure 3.5, la fonction $variate(x)$ va permettre de faire une transition douce entre les valeurs $vmin_a$ et $vmax$ qu'elle va appliquer sur toutes les branches situées en dessous de l'étage $stepmax$ du tronc, et une transition douce entre les valeurs $vmax$ et $vmin_b$ qu'elle va appliquer sur toutes les branches situées au dessus de l'étage $stepmax$ du tronc.

La figure 3.6 illustre quelques résultats obtenus en exécutant plusieurs fois le système représenté par la grammaire 3.2 pour des valeurs de variables spécifiques ($precision = 20$, $stepmax = 17$, $vmin_a = 0.2$, $vmax = 1.0$ et $vmin_b = 0.8$). Ici, les valeurs des différentes variables ont été choisies de manière à ce que la forme des fruits créés soit proche de celle de la tomate mal formée illustrée sur la partie gauche de la figure 3.2 au début de cette section. Comme nous pouvons le voir, grâce à la combinaison des valeurs de variables définies de manière aléatoire et de la fonction contrainte verticale, les fruits résultants de l'exécution du système présentent tous les proportions similaires, c'est-à-dire des branches plus grandes atour du volume de l'étage $stepmax$ et des extrémités plus petites.

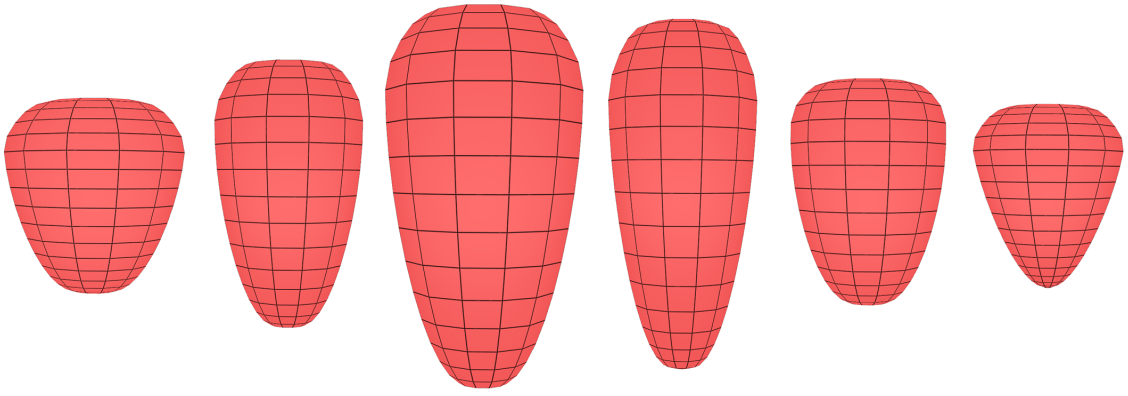


FIGURE 3.6 – Différents résultats de l'exécution du système lorsqu'il est influencé par la fonction de perturbation verticale. Ici les valeurs des variables sont $precision = 20$, $stepmax = 17$, $vmin_a = 0.2$, $vmax = 1.0$ et $vmin_b = 0.8$.

3.3.2.2 Définition de la fonction contrainte horizontale

La fonction contrainte verticale se limite à influencer la taille par défaut des branches en fonction de leur étage au sein du tronc et ne permet pas de les faire varier en fonction du côté du tronc dont elles sont issues. Ainsi, c'est pour compléter la fonction $variate(x)$ que nous avons introduit une fonction contrainte horizontale $hfunction(x)$ qui va permettre cette fois-ci à l'utilisateur d'influencer la taille par défaut des branches en fonction du côté du tronc sur lequel elles seront générées.

Tout comme pour la paramétrisation du système présentée dans le chapitre précédent, l'utilisateur peut choisir une fonction mathématique échantillonnée sur un intervalle bien précis en fonction de ses besoins. Dans la grammaire 3.3 par exemple, nous avons décidé d'utiliser la fonction $f(x) = sin(x)$ qui sera échantillonnée sur l'intervalle $I = [0\,;\pi]$. La grammaire 3.3 montre comment intégrer la fonction contrainte horizontale à notre système et la figure 3.7 illustre le tronc vu du dessus avec la courbe de $f(x)$ échantillonnée tout autour.

Grammaire 3.3 : `Ajout de la fonction contrainte horizontale.`

@Définition des volumes, fonctions et variables@
...

@Fonction contrainte verticale@
$\#define\ stepmax = 13$
$\#define\ vmax = 1$

$\#define\ vmin_a = 0.1$
$\#define\ dx_a = (vmax - vmin_a)/stepmax$
$\#function\ variate_a(x)\ return\ vmin_a + dx_a \times x;$

$\#define\ vmin_b = 0.8$
$\#define\ dx_b = (vmax - vmin - b)/(precision - stepmax)$
$\#function\ variate_b(x)\ return\ vmax - dx_b \times (x - stepmax);$

$\#function\ variate(x)\ return\ x \leq stepmax\ ?\ variate_a(x)\ :\ variate_b(x);$

@Fonction contrainte horizontale@
$\#define\ fmin = 1$
$\#define\ dx = \pi/(ph - 1)$
$\#function\ hfunction(x)\ return\ fmin + sin(dx \times x);$

@Définition de l'axiome et des règles de production@
...
$p03\ A\{\}\{etape < precision\}\{\} \rightarrow A[B(,,,,,<$
$taille(etape) \times variate(etape) \times hfunction(numface - 1) >,,)]_{C*}$
...

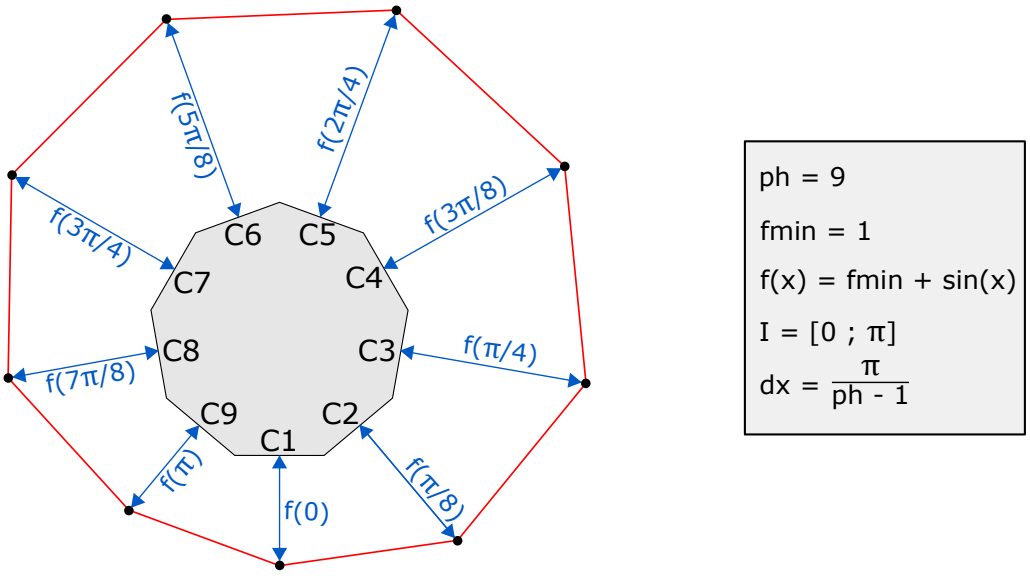


FIGURE 3.7 – Échantillonnage de la fonction sin(x) autour du tronc. En fonction de l'échantillonnage, une valeur de x est assignée à chaque face côté (exemple : $x_{C3} = \pi/4$), permettant ainsi de calculer le facteur correspondant aux branches issues de ces faces.

Dans cette nouvelle version du système, représentée par la grammaire 3.3, nous avons introduit l'utilisation de la fonction contrainte horizontale. Sa paramétrisation fonctionne de la même manière que la paramétrisation autour du tronc présentée dans le chapitre précédent. L'intervalle $I = [0\,;\pi]$ est alors échantillonné en fonction de la précision horizontale (variable ph), qui est égale à 9 dans l'exemple illustré sur la figure 3.7. Ainsi, une fois correctement paramétrée, la fonction $hfunction(x)$ se charge de renvoyer un facteur correspondant à chaque branche en fonction de la face sur laquelle elle est générée. Ce facteur est alors combiné à celui renvoyé par la fonction $variate(x)$ afin de faire en sorte que la forme du fruit respecte les proportions définies par la surface englobante.

La figure 3.8 montre différents résultats obtenus après plusieurs exécutions du système, et pour des paramètres définis sur la figure 3.7. Comme nous pouvons le voir, la taille des branches varie en fonction des faces sur lesquelles elles sont générées, donnant ainsi des branches plus petites lorsque nous nous rapprochons des faces $C1$ et des branches plus grandes lorsque nous sommes près des faces $C5$.

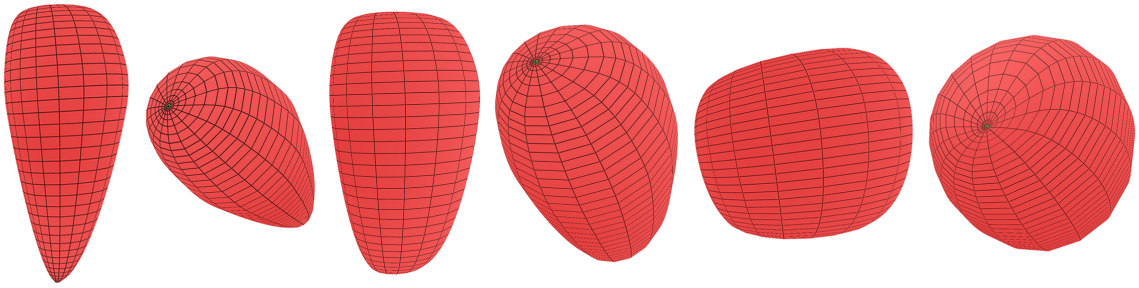


FIGURE 3.8 – Résultats obtenues grâce à l'ajout de la fonction contrainte horizontale. Les valeurs des variables sont celles définies sur la figure 3.7.

3.3.3 Synthèse

Nous avons introduit dans cette section une méthode permettant de faire varier la forme générale d'un fruit de manière aléatoire dans le but de générer des fruits mal formés. Comme nous l'avons vu sur les tomates de la figure 3.2, les malformations semblent avoir des formes aléatoires qui sont difficiles à prédire. Ainsi, pour coller à cet aspect imprévisible, nous avons proposé dans un premier temps d'intégrer de l'aléatoire dans la définition des variables du système, afin de générer des modèles différents à chaque nouvelle exécution. Pour ce faire, nous avons intégré dans notre grammaire deux nouvelles fonctions ($variate(x, taux)$ et $randomval(x, y)$) permettant de changer aléatoirement les valeurs des variables entre deux exécutions du système. Les résultats ainsi obtenus présentaient tous des formes différentes qui, lorsque les valeurs des variables n'étaient pas correctement paramétrées, pouvaient être considérées comme des formes anormales selon les normes définies dans [OEC02] par l'OCDE .

Bien que l'ajout d'une définition aléatoire de variables nous a permis de générer une grande variété de formes de malformations différentes, nous voulions cependant garder un certain contrôle sur la forme du résultat final. Pour ce faire, nous avons ainsi introduit une méthode inspirée de [HJT+13] et qui permet de contraindre la forme finale du fruit à respecter des proportions définies par une surface englobante. La combinaison de l'utilisation de cette surface et de la définition de variables aléatoires nous a ainsi permis de générer des fruits qui ont des formes aléatoires, tout en respectant les proportions définies par les fonctions contraintes verticales et horizontales.

Bien sûr, tout comme la paramétrisation de la forme du fruit présentée dans le chapitre précédent, la force de cette approche réside dans le fait que l'utilisateur n'est pas restreint à l'utilisation des deux fonctions contraintes définies dans cette section. En effet, plutôt que d'utiliser la fonction $sin(x)$ comme fonction contrainte horizontale, rien n'empêche l'utilisateur d'utiliser une autre fonction mathématique permettant de faire varier différemment la taille des branches autour du tronc. En expérimentant avec différentes fonctions contraintes, nous avons ainsi réussi à modéliser les trois tomates mal formées illustrées sur la figure 3.9, et dont les photos ont été tirées de [OEC02].

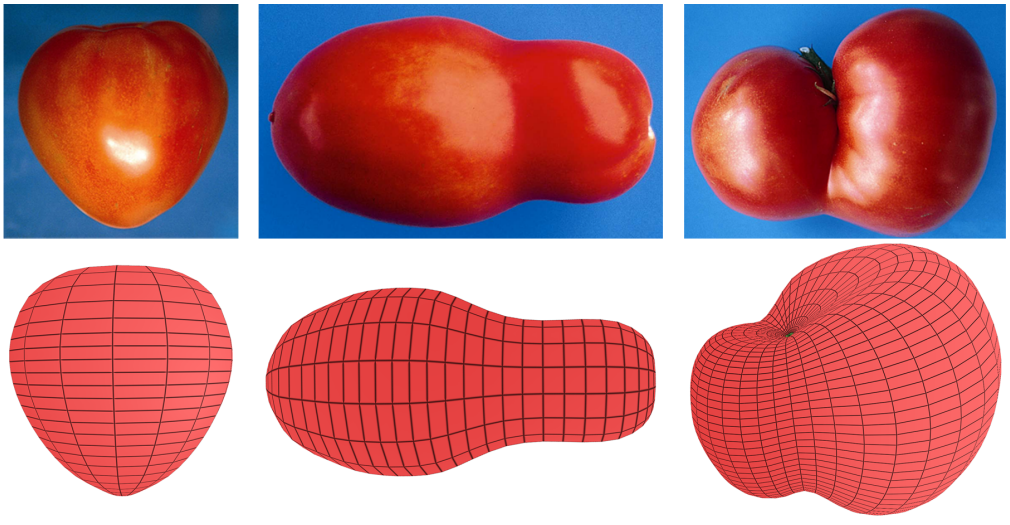


FIGURE 3.9 – (haut) photos tirées de [OEC02] et (bas) modèles 3D correspondants.

3.4 Génération de variations géométriques locales

Maintenant qu'il est possible de créer des variations géométriques globales sur la forme du fruit, nous voulons être capables de représenter des anomalies géométriques à une échelle plus petite. En nous basant sur les normes définies dans [OEC02] et [UNE12], nous pouvons voir que les trois défauts géométriques visibles à la surface d'une tomate qui sont le plus souvent rencontrés sont : les excroissances, les crevasses et les impacts de grêle. Dans cette section, nous allons introduire trois extensions de notre modèle qui nous donnent la possibilité de représenter ces trois types de défauts de surface. Ainsi, dans un premier temps, nous présenterons un mécanisme de hiérarchie des axes et des volumes permettant de cibler des parties bien précises de la surface du fruit dans le but d'y générer des excroissances. Ensuite, nous présenterons deux nouvelles règles de production, l'une permettant de générer des crevasses à la surface du fruit, et l'autre nous donnant la possibilité d'y représenter des impacts de grêle de manière aléatoire.

3.4.1 Génération d'excroissances

Chez certaines tomates, il est possible d'observer la présence d'excroissances le long de leur surface qui, à l'instar des variations géométrique globales, ont tendance à apparaître lorsque les conditions climatiques ne sont pas favorables à la croissance du fruit. Au cours de nos travaux, nous nous sommes intéressés à deux types d'excroissances parmi les plus courants : les excroissances proches du pédoncule du fruit, illustrées à gauche sur la figure 3.10, et les excroissances stylaires, situées à l'opposé du pédoncule et illustrées sur la partie droite de la même figure.

Selon Blancard *et al.* [Bla09], les excroissances stylaires, aussi appelées mucrons, seraient dues à l'absence de chute des pétales et des étamines de la fleur au début de la phase de formation de la tomate, lui conférant ainsi un aspect proche du citron. Quant aux excroissances apparaissant non loin du pédoncule du fruit, elles sont appelées appendices et seraient généralement dues au froid et à une humidité de l'air qui est trop importante au cours du développement de la tomate.

FIGURE 3.10 – (gauche) Des excroissance proches du pédoncule, aussi appelées appendices et (bas) des excroissances stylaires, aussi appelées mucrons.

3.4.1.1 Hiérarchie et sélection de volumes

Afin de générer des excroissances à la surface de nos fruits, nous voulons pouvoir cibler des volumes spécifiques au sein de la structure arborescente générée par notre système, dans le but de pouvoir y créer des volumes représentant ces anomalies. Pour ce faire, nous nous servons d'une notion de hiérarchie des volumes introduite par Terraz *et al.* , à laquelle nous avons ajouté une extension basée sur l'utilisation de chemins.

Ordre d'axe et d'étage d'un volume dans l'axe.
La notion de hiérarchie, introduite dans [TGM+09], est représentée par deux paramètres qui sont assignés à chaque volume du système : l'ordre de l'axe auquel il appartient, et l'étage au sein duquel il se trouve dans cet axe. Ces deux paramètres sont initialisés lors de la création d'un volume et les valeurs qui leurs sont attribuées dépendent de celles des paramètres du volume générateur ainsi que du label de la face génératrice.

Par exemple, si nous prenons l'axiome de départ, celui-ci se trouve à l'étage 0 de l'axe d'ordre 0 de la structure arborescente. Par la suite, les valeurs des paramètres de chaque volume créé seront définies de la manière suivante. Si un volume B est créé sur la face E d'un volume A par exemple, alors il appartient au même axe que A et nous n'incrémentons que la valeur de son étage. Si A est l'axiome du système, alors B se trouve à l'étage 1 de l'axe d'ordre 0. Par contre, si un volume C est créé sur une face latérale de ce même volume B, alors C formera un nouvel axe et il faudra ainsi incrémenter la valeur de l'axe et réinitialiser à 0 la valeur de l'étage de C. Nous dirons alors que C se trouve à l'étage 0 d'un axe d'ordre 1. En se basant sur les notions d'axes et d'étages, nous pouvons dire que dans notre structure arborescente, tous les volumes A qui forment le tronc sont donc des volumes de l'axe d'ordre 0, alors que toutes les branches B appartiennent à des axes d'ordre 1. La figure 3.11 permet de mieux comprendre ce concept.

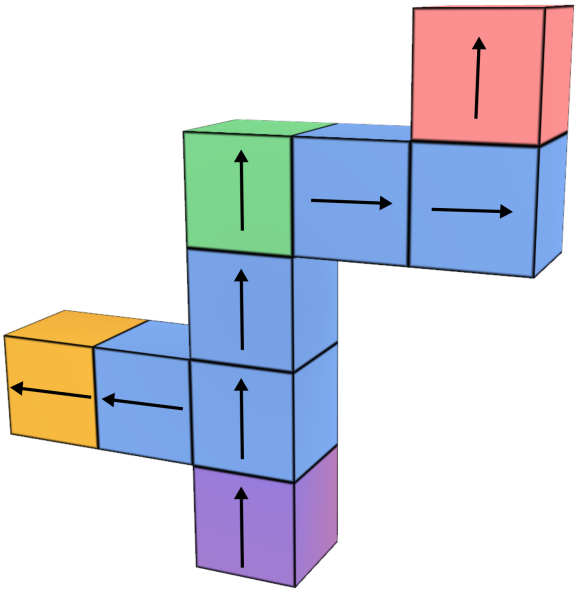


FIGURE 3.11 – Hiérarchie. Les flèches pointent vers la face E de chaque volume. Le volume violet est l'axiome, qui se trouve à l'étage 0 de l'axe d'ordre 0. Le volume vert est à l'étage 3 du même axe. Le volume jaune se trouve à l'étage 1 de l'axe d'ordre 1, et le rouge est à l'étage 0 de l'axe d'ordre 2.

Notion de chemin.
Bien que l'ajout des notions d'axes et d'étages nous permet de différencier des ensembles de volumes, il nous est impossible de pouvoir sélectionner un volume spécifique de la structure arborescente uniquement à l'aide de ces deux paramètres. Ainsi, pour palier ce problème, nous avons introduit une extension permettant de définir les coordonnées exactes d'un volume au sein de la structure arborescente. Cette extension, appelée "chemin", est définie par une suite de caractères qui est attribuée à chaque volume lors de sa création. Le chemin d'un volume peut ainsi être vu comme un historique permettant de retracer l'enchaînement des différentes règles d'ajout qui ont eu lieu depuis l'axiome de départ et ce, jusqu'à la création de ce volume. Voici un exemple de règle d'ajout ciblant un volume A en fonction de son chemin :

$$A(3, C1, 4, C5, 2) \rightarrow A[B]_E \tag{3.1}$$

Dans cet exemple, la règle de production 3.1 ne s'appliquera qu'à un seul et unique volume A qui se trouve aux coordonnées indiquées par le chemin qui est contenu entre les parenthèses. Pour pouvoir retrouver le volume A qui est censé être la cible de la règle 3.1, nous devons lire son chemin $(3, C1, 4, C5, 2)$ de la manière suivante :

1. En partant de l'axiome, se déplacer jusqu'au volume situé au troisième étage de l'axe.
2. Se déplacer sur le volume ayant été généré sur la face $C1$ du volume courant.
3. Depuis le volume courant, aller au volume situé au quatrième étage de son axe.
4. Se déplacer sur le volume ayant été généré sur la face $C5$ du volume courant.
5. Monter de deux étages dans l'axe du volume courant.
6. Si le volume courant est bien un volume A, il s'agit du bon volume.

Comme nous pouvons le voir, un chemin est un parcours composé d'une succession de translations et de rotations permettant de retrouver un volume spécifique en partant de l'axiome de départ. Ainsi, sur la structure illustrée sur la figure 3.11 par exemple, le chemin permettant d'atteindre le volume rouge est $(3, C1, 1, C2, 0)$, celui qui correspond au volume vert est (3) et celui qui mène au volume jaune est $(1, C3, 1)$.

Sélection d'un ensemble de volumes.
En plus de pouvoir sélectionner un volume spécifique de la structure arborescente, l'utilisation des chemins nous donne également la possibilité de sélectionner un ensemble de volumes. Pour ce faire, nous pouvons utiliser $C*$ pour indiquer que nous souhaitons parcourir tous les sous-axes du volume courant, le symbole $*$ au milieu du chemin pour parcourir tous les étages de l'axe, et ce même symbole à la fin du chemin pour récupérer tous les volumes du dernier axe rencontré dans le parcours. Ainsi, si nous prenons la structure arborescente de la figure 3.12 par exemple, voici quelques exemples de chemins pouvant récupérer des ensembles de volumes bien précis :

- (0) permet de sélectionner l'axiome de départ, de couleur violet sur la figure.
- $(3, C1, *)$ permet de récupérer tous les volumes verts.
- $(3, C1, *, C*, *)$ cible tous les volumes rouges de la structure arborescente.
- $(1, C3, *, C1, 1)$ récupère l'ensemble des volumes jaunes.

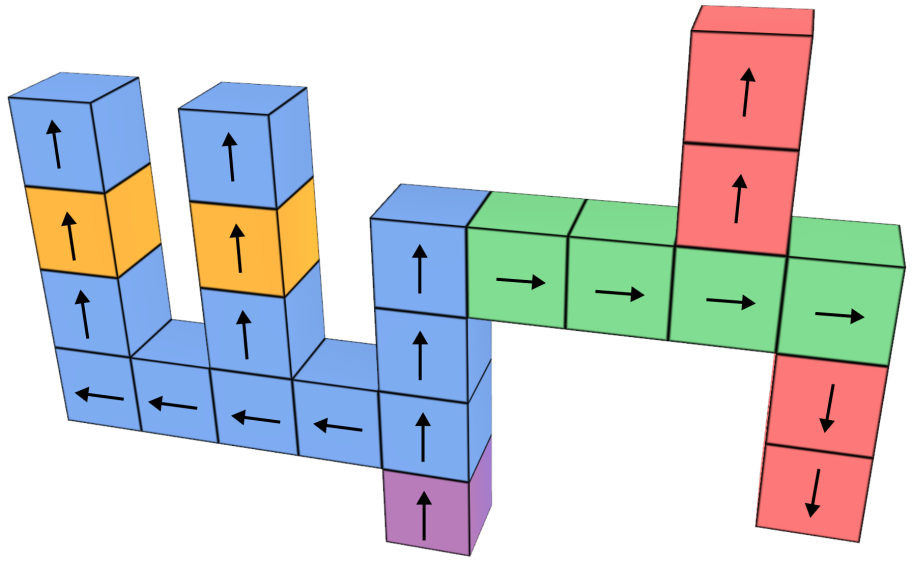


FIGURE 3.12 – Sélection d'un ensemble de volumes. Ici les flèches pointent vers les faces E des volumes.

3.4.1.2 Résultats

Grâce à l'extension de la notion de hiérarchie au sein de notre modèle, il nous est maintenant possible d'agir localement sur la surface de nos fruits dans le but d'y générer des excroissances. Ainsi, comme nous pouvons le voir sur la figure 3.13 par exemple, nous avons pu modéliser des tomates présentant des variations locales similaires à celles des fruits de la figure 3.10, notamment grâce à l'utilisation de chemins pour cibler des volumes précis de la structure arborescente.

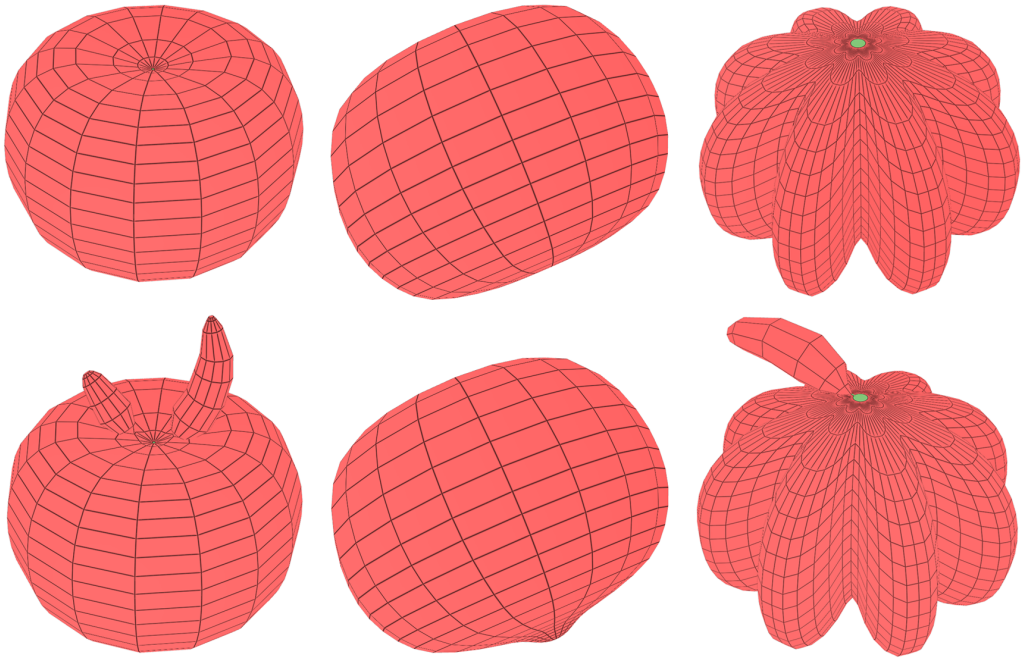


FIGURE 3.13 – Résultats montrant des excroissances générées à l'aide de l'utilisation de chemins.

3.4.2 Génération de crevasses

Le fruit est un objet très fragile dont l'intégrité peut très vite être perturbée par des facteurs extérieurs. En effet, comme nous l'avons vu au début de cette section, le fruit étant le résultat d'un enchaînement de processus qui interagissent fortement entre eux, sa forme finale est souvent sujette à des modifications lorsque l'un de ces processus ne se déroule pas correctement. Ces modifications se matérialisent alors par des défauts globaux ou locaux en fonction de la nature du phénomène qui a perturbé le processus de croissance. Parmi la grande variété de défauts géométriques locaux que nous pouvons observer chez les tomates, les crevasses font parties de ceux que l'on rencontre le plus souvent.

Selon l'OCDE [OEC02], l'apparition de crevasses serait généralement due soit à une manipulation brutale du fruit lors de sa récolte ou soit à une croissance trop rapide provoquée par des irrégularités climatiques (exemple : une forte pluie en période de sécheresse). En fonction des caractéristiques qu'elle présente, une crevasse affectera plus ou moins la qualité commerciale d'une tomate. Ainsi, selon [OEC02], si un fruit présente une crevasse à sa surface, alors il ne pourra être considéré comme étant propre à la consommation que si cette dernière ne dépasse pas trois centimètres de longueur, n'est pas trop profonde et est complètement cicatrisée.

Au cours de nos recherches, nous avons identifié trois types de crevasses : les crevasses concentriques formant des cercles autour du pédoncule, les crevasses radiales (ou longitudinales) partant du pédoncule pour se diriger vers l'autre bout de la tomate, et les crevasses artificielles de type transversales. En haut de la figure 3.14, nous pouvons voir quelques tomates crevassées qui, selon [OEC02] peuvent être commercialisées, et la partie basse montre des tomates crevassées qui ne peuvent pas être commercialisées, notamment à cause de crevasses qui sont trop profondes ou non cicatrisées.

FIGURE 3.14 – (Haut) Tomates aptes à être commercialisées : (Gauche à droite) une crevasse concentrique et deux crevasses longitudinales. (Bas) Tomates impropres à la consommation : (gauche à droite) une crevasse concentrique profonde, une crevasse artificielle fraîche due à une manipulation brutale et une crevasse longitudinale profonde.

3.4.2.1 Opération de rupture du bord

Nous voulons pouvoir générer des crevasses à la surface de nos fruits. Pour ce faire, nous avons introduit dans le modèle des 3G-cartes L-systèmes une nouvelle opération appelée : opération de rupture du bord. Dans les modèles 3D de fruits générés par notre système, les différentes couches du péricarpe sont matérialisées par des volumes qui sont collés les uns aux autres. Au niveau de la 3G-carte sous-jacente, ces liaisons inter-volumes se traduisent par des involutions de type α_3. Ainsi, tout comme ce qui se passe à la surface des vrais fruits, cette opération va simuler la rupture des différentes couches en rompant les liaisons α_3 qui existent entre des volumes spécifiques, afin de les séparer et de faire apparaître une crevasse entre eux. Pour appliquer cette opération, nous utilisons la règle de production suivante :

$$A \rightarrow *A_{C2}\_B_{C4} * (\ epaisseur,\ longueur,\ profondeur,\ cicatrices\ ) \qquad (3.2)$$

Dans cet exemple, la règle de production 3.2 va créer des crevasses au niveau de tous les volumes A de la surface du fruit, dont la face $C2$ est adjacente à la face $C4$ d'un volume B, appartenant lui aussi à la surface. Pour ce faire, le système va rompre toutes les liaisons de type α_3 qui existent entre ces deux faces et va déplacer les sommets de la surface de manière à créer une crevasse définie par les paramètres entre parenthèses. Dans notre modèle, une crevasse est caractérisée par cinq paramètres bien précis qui, selon [OEC02], suffisent à définir si une tomate sera propre à la consommation ou pas.

Direction de la crevasse.
Comme nous l'avons vu sur la figure 3.14, il existe trois types de crevasses différentes (longitudinales, transversales et concentriques) définies en fonction de la direction de la crevasse. Dans une règle de rupture du bord, cette direction est définie par l'adjacence des deux volumes de la règle. Ainsi, dans la règle 3.2 par exemple, la crevasse se propagera le long des faces A_{C2} et B_{C4} lorsque leurs relations d'adjacence seront rompues.

Épaisseur de la crevasse.
Le premier paramètre entre parenthèses de la règle 3.2 permet de définir la taille de l'ouverture de la crevasse. Selon nos observations, il est rare de rencontrer une tomate dont la crevasse est très épaisse. Cela pourrait s'expliquer par le fait que, étant donné que l'endocarpe d'une tomate est visqueux, une ouverture trop importante d'une crevasse aurait sûrement tendance à vider la tomate de son "jus", accélérant ainsi le pourrissement du fruit. La figure 3.15 montre l'influence de ce paramètre sur le résultat final.

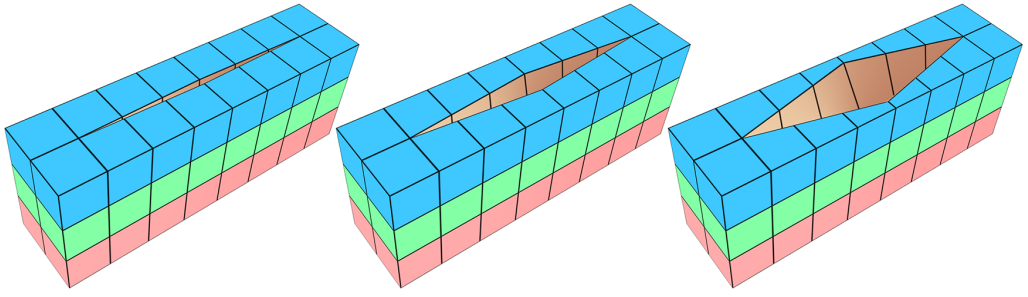


FIGURE 3.15 – Création de crevasses. De gauche à droite : augmentation de l'épaisseur.

Longueur de la crevasse.
Comme expliqué au début de cette section, dans les normes définies dans [OEC02], la longueur d'une crevasse joue un rôle important dans la qualité d'un fruit. Ainsi, plus la longueur d'une crevasse sera élevée, moins la tomate aura de chances d'être considérée comme étant propre à la consommation. Dans la règle de rupture du bord, le deuxième paramètre entre parenthèses permet de définir la longueur de la crevasse, et son influence est illustrée sur la figure 3.16. Ainsi, plus sa valeur sera élevée, plus la crevasse se propagera le long de la direction définie par la jonction des deux volumes A et B.

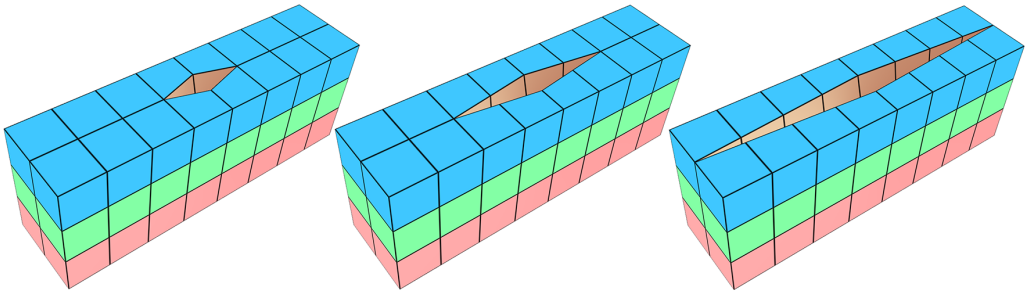


FIGURE 3.16 – Création de crevasses. De gauche à droite : augmentation de la longueur.

Profondeur de la crevasse.
Le troisième paramètre entre parenthèses permet de définir la profondeur de la crevasse. Plus la valeur de ce paramètre sera élevée, plus la plaie formée par la règle de rupture des bords se propagera en profondeur, révélant ainsi les couches internes de notre fruit. Bien qu'il soit rare de rencontrer des tomates présentant des crevasses très profondes, allant par exemple jusqu'à l'endocarpe, l'utilisateur est libre d'attribuer n'importe quelle valeur au paramètre de profondeur. La figure 3.17 montre comment les valeurs de ce paramètres influent sur le résultat final.

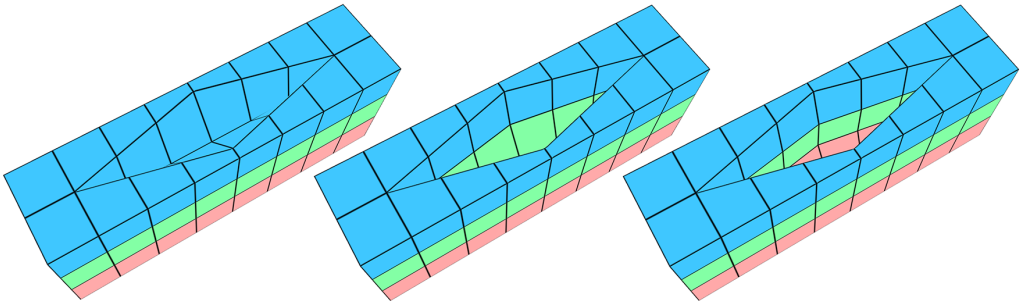


FIGURE 3.17 – Création de crevasses. De gauche à droite : augmentation de la profondeur.

Niveau de cicatrisation.
Selon [OEC02], toute tomate comportant une crevasse qui n'est pas cicatrisée aura tendance a être plus facilement victime de maladies et ne peut donc pas être considérée comme étant propre à la consommation. Ainsi, bien que les travaux présentés dans ce mémoire ne s'attardent pas sur l'aspect rendu des fruits et de leurs défauts, il est néanmoins important de pouvoir définir si une crevasse générée par notre modèle est cicatrisée

ou pas. Pour ce faire, nous avons attribué un marqueur à chaque face censée représenter un tissu cicatrisé dans le but de lui assigner une couleur appropriée (marron sur nos figures). Bien sur, ce marqueur pourra par la suite être utilisé plus tard au sein d'un moteur de rendu spécialisé afin d'attribuer à ces faces un matériau réaliste spécifique. Dans notre règle de rupture, le dernier paramètre entre parenthèses permet de définir si chaque couche atteinte par une crevasse a été cicatrisée ou non. Ainsi, si ce paramètre est égal à 2 par exemple, alors seules les deux couches les plus proches du bord seront cicatrisées. Si par contre ce paramètre est égal au symbole $*$, alors la crevasse sera totalement cicatrisée. La figure 3.18 montre l'influence de ce paramètre sur le résultat final.

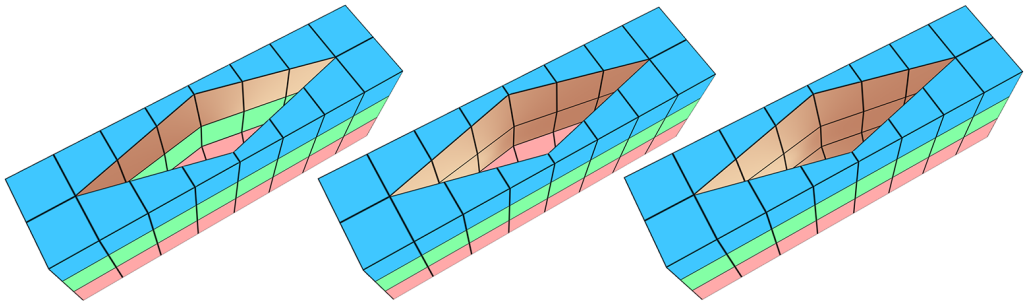


FIGURE 3.18 – Création de crevasses. De gauche à droite : valeur du niveau de cicatrisation égale à 1, 2 et 3.

3.4.2.2 Résultats

Comme nous pouvons le voir sur la figure 3.19, grâce à l'introduction de la règle de rupture du bord, nous avons généré trois tomates présentant chacune un type de crevasse spécifique : une crevasse longitudinale, une crevasse transversale et une crevasse concentrique. Pour ce faire, nous avons ciblé des volumes adjacents et qui appartiennent à la surface du fruit grâce à leurs chemins afin de définir une direction, que nous avons combiné aux différents paramètres de la règle de rupture du bord dans le but de donner la forme voulue aux différentes crevasses que nous avons généré. Nous noterons cependant que, étant donné le fait que les crevasses que nous pouvons trouver sur des vraies tomates sont rarement très ouvertes, nous avons jugé suffisant de limiter l'épaisseur d'une crevasse à la taille des volumes qui l'entourent. Ainsi, une extension possible de cette opération pourrait être l'introduction de la prise en compte des voisins éloignés de la crevasse.

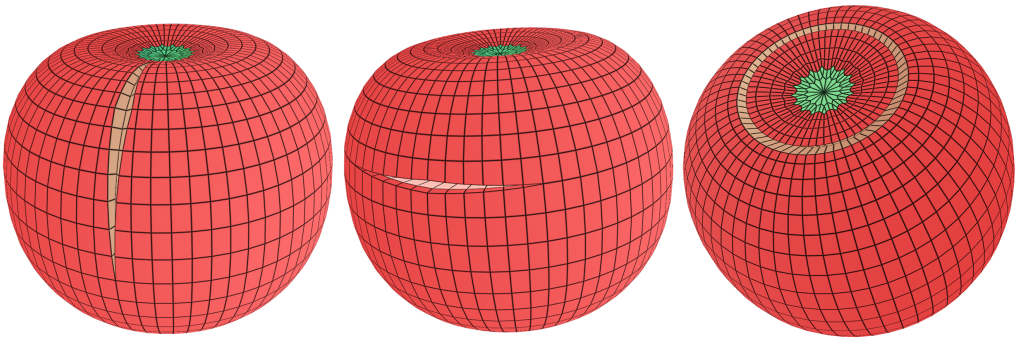


FIGURE 3.19 – Tomates crevassées générées à l'aide de notre modèle. Gauche à droite : une crevasse longitudinale cicatrisée, une crevasse transversale non cicatrisée et une crevasse concentrique cicatrisée.

3.4.3 Impacts de grêle

Au cours de la transition entre l'hiver et le printemps, le temps devient très changeant et signe souvent l'arrivée de pluies importantes, accompagnées de grêle. Dans les cultures de tomates de plein champ, c'est-à-dire des cultures en plein air qui ne sont pas protégées des intempéries, il n'est pas rare d'observer des dégâts plus ou moins importants qui sont causés par la grêle. Les fruits étant particulièrement vulnérables aux grêlons, lorsque ces derniers les percutent, ils créent ainsi aux points d'impact des éclatements plus ou moins prononcés. Les blessures occasionnées par ces impacts se transforment alors en points d'entrée facilitant la pénétration de bactéries induisant des pourritures. Selon [OEC02], le fait que la grêle crée souvent des dégâts importants à la surface des fruits et facilite grandement leur infection fait que toute tomate ayant été victime d'une attaque de grêle ne peut pas être considérée comme étant propre à la consommation. Nous pouvons voir sur la figure 3.20 quelques exemples de tomates ayant été attaquées par la grêle.

FIGURE 3.20 – Tomates endommagées par des impacts de grêle.

3.4.3.1 Opération d'impact

Afin de pouvoir générer des impacts de grêle à la surface de nos fruits, nous avons introduit dans notre modèle une nouvelle opération appelée : opération d'impact. Cette opération, inspirée des opérations topologiques de Bézin *et al.* (voir [BCS+11]), va simuler un impact à la surface du fruit en ciblant un point et en le tirant vers l'intérieur du modèle 3D le long de son arête dans le but de créer un creux. Comme nous pouvons le voir sur la figure 3.21, cette opération est simplifiée grâce à la régularité des modèles 3D obtenus par notre méthode. De plus, grâce à cette régularité, l'implémentation de cette opération s'en trouve simplifiée car il nous suffit alors de parcourir les liaisons α_3 des volumes pour passer d'une couche à l'autre. Bien sûr, si cela s'avère nécessaire dans la suite de nos travaux, elle pourra être généralisée à une topologie plus complexe comme ces le cas dans [BCS+11].

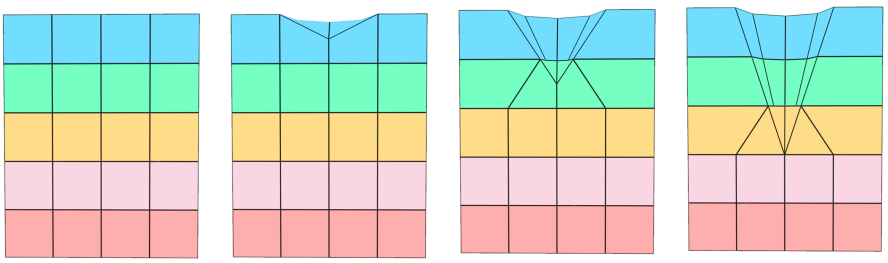


FIGURE 3.21 – Opération d'impact. Nous pouvons voir de gauche à droite ce qui se passe lorsqu'on augmente la valeur du paramètre profondeur.

Pour pouvoir utiliser l'opération d'impact dans un 3G-carte L-système, l'utilisateur peut faire appel à la règle d'impact, qui lui permet de choisir un point précis de la surface afin d'y créer un creux pour simuler un impact de grêle. Voici un exemple de règle d'impact :

$$A \rightarrow \backslash\ A\_B\_C\_D\ /(\ profondeur,\ cicatrisation,\ arret\ ) \tag{3.3}$$

Dans cet exemple, lorsque la règle 3.3 est appliquée, notre système récupère dans un premier temps tous les points qui sont situés à la surface du fruit et qui sont partagés par des volumes de label A, B, C et D. Ensuite, une fois que tous ces points ont été localisés, ils sont déplacés le long de l'arête partagée par les quatre volumes afin de former un creux. Dans notre modèle, le creux généré par une règle d'impact est défini selon trois paramètres.

Profondeur.
Le paramètre $profondeur$ permet de définir la distance que doit parcourir le point d'impact le long de son arête. Comme nous pouvons le voir sur les figures 3.21 et 3.22, plus la valeur du paramètre $profondeur$ augmente, plus le creux devient profond et révèle ainsi les différentes couches internes du modèle 3D.

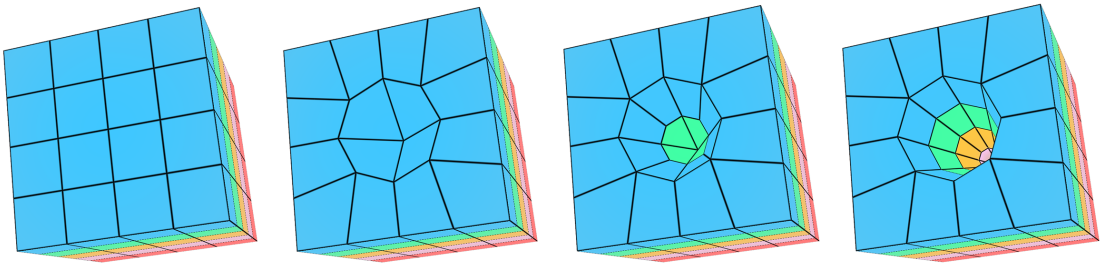


FIGURE 3.22 – Opération d'impact. Augmentation du paramètre de profondeur de la règle d'impact.

Niveau de cicatrisation.
Comme pour les crevasses, bien que nos travaux ne se focalisent pas sur l'aspect rendu des fruits et de leurs défauts, nous avons introduit au sein de notre règle d'impact, un moyen de définir si les différentes couches atteintes par un creux sont cicatrisées ou pas. Ainsi, un paramètre $cicatrisation$ de valeur égale à 1 par exemple permet de marquer les parois de la couche la plus externe comme étant cicatrisées, alors qu'une valeur égale à 0 définie une plaie fraîche, non cicatrisée.

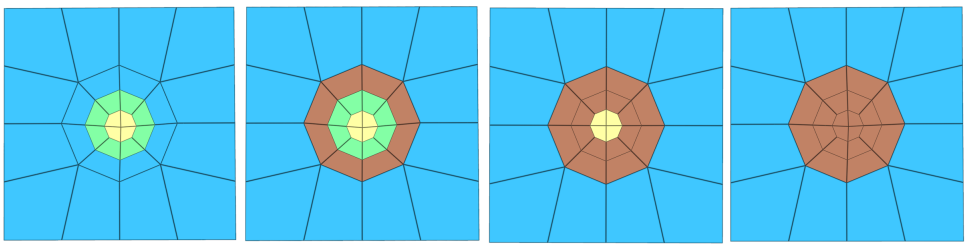


FIGURE 3.23 – Opération d'impact. Gauche à droite : paramètre de cicatrisation égal à 0, 1, 2 et 3 (ou $*$).

Point d'arrêt.
Le paramètre $arret$ de la règle d'impact permet de contraindre la propagation d'un creux à s'arrêter si certaines conditions sont rencontrées. Ainsi, si au cours de son déplacement le point d'impact rencontre un volume dont le label est égal à celui défini par $arret$, alors le creux ne se propagera pas au delà de ce volume. L'utilisation de ce paramètre permet de garder un contrôle sur la création de creux à la surface de nos fruits. En effet, si notre système permet par exemple de générer une tomate de forme différente à chaque exécution, alors l'appel à une même règle d'impact sur tous ces fruits ne donnera pas toujours le même résultat. Ainsi, en donnant pour point d'arrêt l'endocarpe d'un fruit 3D par exemple, alors les creux générés à sa surface ne dépasserons jamais cette couche du péricarpe, même si la valeur de leur paramètre de profondeur est élevée. Nous pouvons voir par exemple sur la figure 3.24 que, lorsque le paramètre $arret$ interdit de dépasser les volumes jaunes, alors le creux s'élargit à sa base, mais ne dépasse jamais ce point d'arrêt.

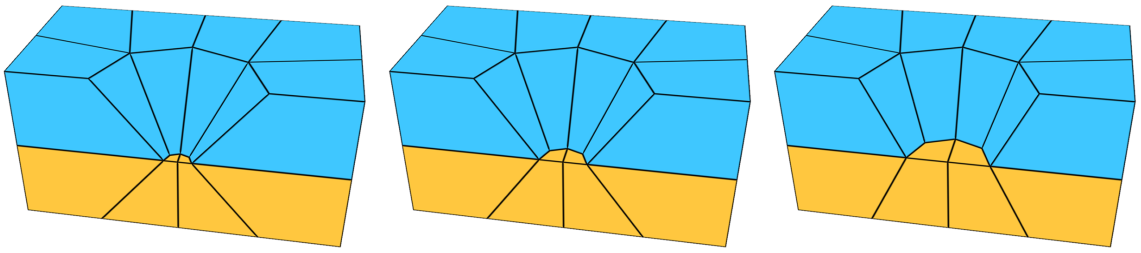


FIGURE 3.24 – Opération d'impact. Gauche à droite : paramètre de cicatrisation égal à 0, 1, 2 et 3 (ou $*$).

3.4.3.2 Règle de génération aléatoire d'impacts et résultats

Dans le but de reproduire le caractère aléatoire des dégâts pouvant être causés par une averse de grêle, nous avons introduit en plus de la règle d'impact ciblée, une règle de production permettant de générer de manière aléatoire des impacts de grêle à la surface d'un fruit. Voici le prototype d'une règle de génération aléatoire d'impacts.

$$B \rightarrow \backslash B/(\ profmin,\ profmax,\ cicatrisation,\ arret,\ quantite\ ) \tag{3.4}$$

Dans cet exemple, l'application de la règle 3.4 va permettre de générer sur une surface composée de volumes B un nombre d'impacts de grêle égal à la valeur de la variable $quantite$. De plus, la position de chacun de ces impacts sera déterminé de manière aléatoire, et sa profondeur sera comprise entre les valeurs des variables $profmin$ et $profmax$. Quant aux variables $cicatrisation$ et $arret$, elles remplissent le même rôle que les variables portant le même nom dans la règle d'impact "classique".

Comme nous pouvons le voir sur la figure suivante, l'utilisation d'une telle règle nous donne la possibilité de choisir un niveau de dégât pour chacun de nos fruits, sans avoir à nous soucier du placement de chaque impact de grêle.

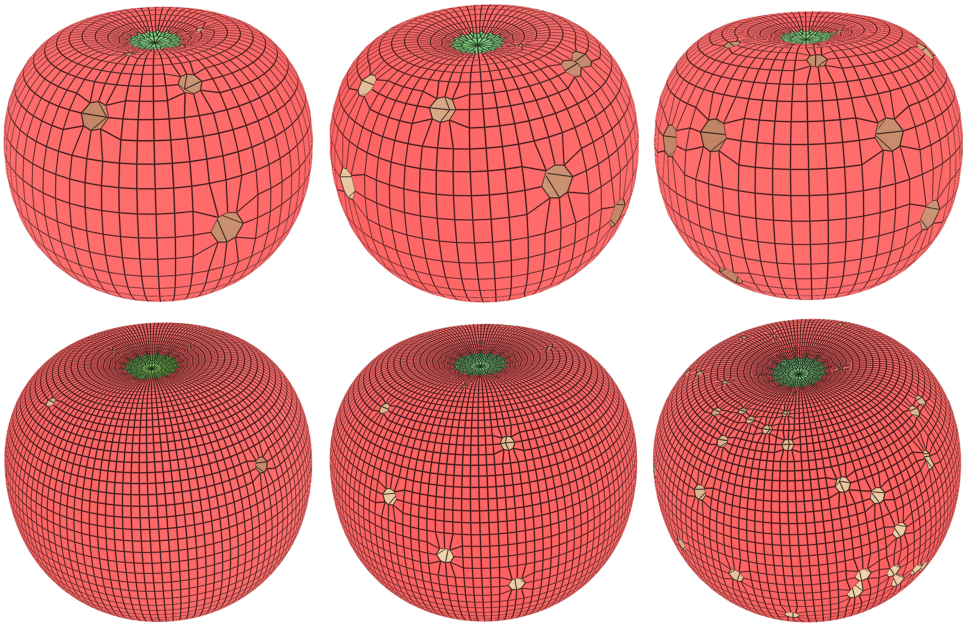


FIGURE 3.25 – Opération d'impact. Gauche à droite : paramètre de cicatrisation égal à 0, 1, 2 et 3 (ou $*$).

3.4.4 Rétrécissement et propagation de maladies au sein du fruit

Comme nous l'avons vu dans le premier chapitre de ce mémoire, la fonction d'un fruit évolue tout au long de son cycle de vie. Ainsi, une fois que la fécondation de la fleur est terminée, un ensemble d'étapes se succèdent au cours desquelles son premier rôle est de protéger les ovules pendant que ces derniers se transforment en graines. Ensuite, une fois que la phase d'expansion cellulaire est terminée, s'en suit alors une étape de maturation au cours de laquelle la nouvelle fonction du fruit est de favoriser la reproduction de l'espèce en propageant les graines fraîchement formées. Pour ce faire, il change alors de couleur, devient plus sucré et plus juteux afin d'inciter les animaux à l'ingérer pour pouvoir ensuite disséminer les graines dans la nature au travers de leurs déjections. Cependant, dans certains cas, le fruit n'arrivera pas à se faire manger. Pour pouvoir malgré tout disséminer les graines, il va alors entrer dans une phase de sénescence, au cours de laquelle il va se décomposer petit à petit, afin de les libérer pour qu'elles puissent se répandre. La figure 3.26 montre des photos d'une tomate qui pourrie au fil des jours, pendant trois mois.

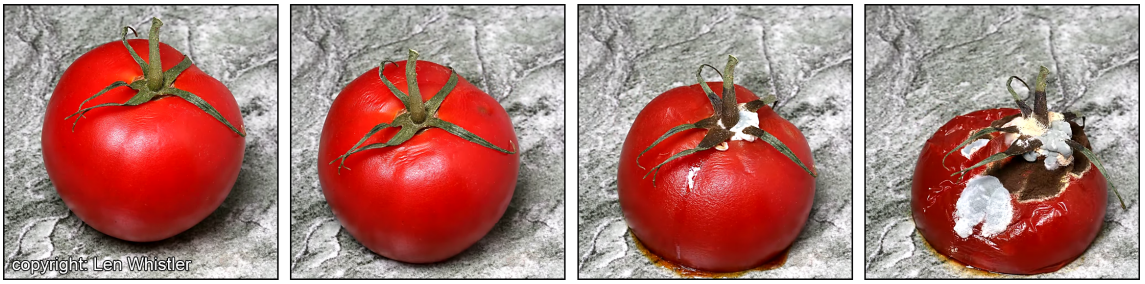


FIGURE 3.26 – Sénescence du fruit. Photos d'une tomate qui s'affaisse au fil du temps.

3.4.4.1 Opération de rétrécissement

Le fruit est un être vivant. Comme tout être vivant, il est principalement composé d'eau, et la teneur en eau varie d'une espèce à l'autre. Dans le cas de la tomate par exemple, lorsque le fruit atteint sa maturité, il est généralement composé d'environ 94% d'eau (voir [BH94]). Cependant, tout au long de son cycle de vie, le fruit perd cette eau par transpiration, ce qui a pour conséquence le rétrécissement de son volume (figure 3.26).

Règle de rétrécissement.
Dans le but de pouvoir simuler ce phénomène d'affaissement du fruit due à la transpiration, nous avons introduit une notion de rétrécissement au concept de 3G-cartes L-systèmes. Dans notre modèle, il est possible de faire vieillir les volumes de la structure arborescente générée par notre système paramétrique grâce à l'opération de rétrécissement. Comme son nom l'indique, cette opération va permettre simuler la perte d'eau des cellules du fruit en les faisant rétrécir, à l'instar des travaux présentés dans [KRB11] par Kider *et al.* . Pour ce faire, nous utilisons la règle de rétrécissement suivante :

$$B \rightarrow ((B\ ,\ 0.1)) \tag{3.5}$$

Dans cet exemple, à chaque fois que nous appliquerons la règle 3.5 à notre système, tous les prismes B de la structure arborescente perdrons 10% de leur volume. Dans notre modèle, pour représenter la perte de 10% du volume d'un prisme, une réduction de 10% est appliquée à toutes les arêtes qui le composent. Si l'utilisateur souhaite faire vieillir tous les volumes du système, il lui suffit d'écrire $((*\ ,\ 0.1))$. La figure 3.27, montre ce qui se passe lorsqu'on applique deux fois la règle de rétrécissement à un modèle composé de deux volumes adjacents.

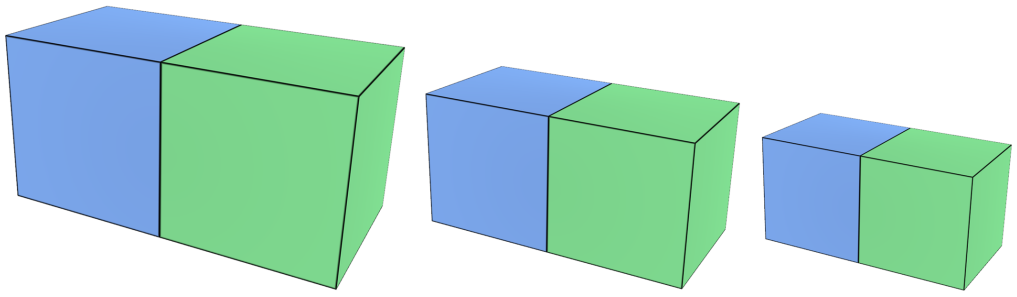


FIGURE 3.27 – Deux applications successives de la règle de rétrécissement à deux volumes adjacents.

Taux de transpiration.
Comme nous pouvons le voir sur la figure 3.27, la règle de rétrécissement telle qu'elle est implémentée ne présente que peu d'intérêt car elle ne simule qu'un simple rétrécissement du modèle entier. Ainsi, afin d'augmenter le contrôle sur le processus de rétrécissement, nous avons ajouté à chaque volume un nouveau paramètre K qui représente son taux de transpiration. En effet, les différents éléments de la structure interne d'un fruit ne présentant pas tous les mêmes propriétés de matériau, ils auront ainsi tendance à perdre au fil du temps une quantité d'eau qui leur est propre. Ainsi, plus la valeur de ce paramètre sera

élevée pour un volume donné, plus ce dernier diminuera rapidement à chaque application de la règle de rétrécissement. L'utilisation du paramètre K peut par exemple s'avérer utile lorsque nous voulons différencier les graines des couches du péricarpe. En effet, étant donné le fait que les graines d'un fruit auront tendance à perdre extrêmement peu de volume au fil du temps, les volumes de notre système qui sont censés les représenter auront alors un paramètre K de valeur proche de zéro. La figure 3.28 montre ce qui se passe sur les deux volumes adjacents de la figure 3.27 si nous leur attribuant des paramètres K de valeurs différentes.

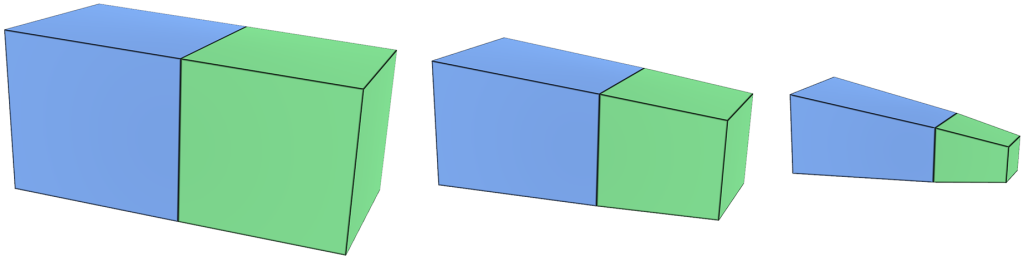


FIGURE 3.28 – Influence du taux de transpiration K. Ici le volume vert a une valeur de K supérieure à celle du volume bleu.

3.4.4.2 Accélération locale du rétrécissement

Comme on peut le voir sur la figure 3.26, le rétrécissement de la tomate n'est pas régulier tout le long de sa surface. En effet, nous pouvons remarquer par exemple que l'affaissement est particulièrement rapide au niveau de la zone de contact entre le fruit et le sol, ainsi qu'au niveau du pédoncule après que celui-ci ait été infecté par des bactéries. Cette accélération du processus de rétrécissement s'explique par le fait que, lorsque le fruit est victime de maladies, sa structure interne a tendance à se fragiliser très rapidement, ce qui se ressent sur un affaissement accéléré du fruit.

Pour simuler ce phénomène d'accélération du rétrécissement, nous avons introduit une règle de pourrissement ciblé. Voici un exemple d'utilisation de cette règle :

$$A \rightarrow ((A,\ B,\ C,\ D\ ,\ 2.5,\ 2,\ 1.5)) \tag{3.6}$$

Dans cet exemple, à l'application de la règle 3.6, le système va récupérer dans un premier temps tous les sommets de la surface du fruit qui sont partagés par quatre volumes de labels A, B, C et D. Ensuite, il va appliquer un masque de coefficients, similaire à un masque de convolution, qui va influencer tous les paramètres K des volumes adjacents à ces sommets dans un rayon de trois volumes par trois volumes. Sur la figure 3.29, nous pouvons voir comment un tel masque est appliqué autour du point blanc de la tomate. Les volumes bleus sont ceux dont le paramètre K va être multiplié par le premier coefficient de la règle de pourrissement ciblé (2.5 dans la règle 3.6), les paramètres K des volumes jaunes vont être multipliés par le second coefficient (2 dans notre exemple) et les volumes verts verront leur taux de transpiration multiplié par le troisième coefficient, soit 1.5 dans la règle 3.6.

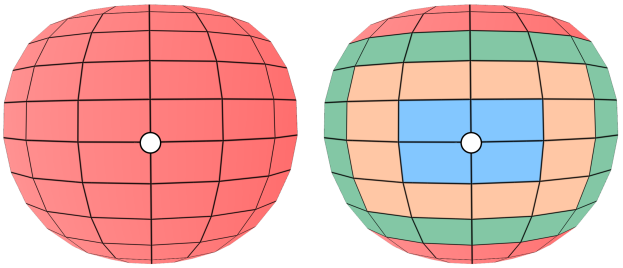


FIGURE 3.29 – Application d'un masque de coefficient pour perturber l'affaissement d'un fruit.

3.4.4.3 Résultats

Grâce à l'introduction de la perte de volume, nous pouvons maintenant simuler le phénomène d'affaissement qu'il est possible d'observer chez les vrais fruits. Ainsi, en utilisant la règle de rétrécissement couplée aux taux de transpiration assignés à chaque volume, nous avons réussi à simuler un rétrécissement régulier sur le modèle 3D de tomate illustré sur la figure suivante.

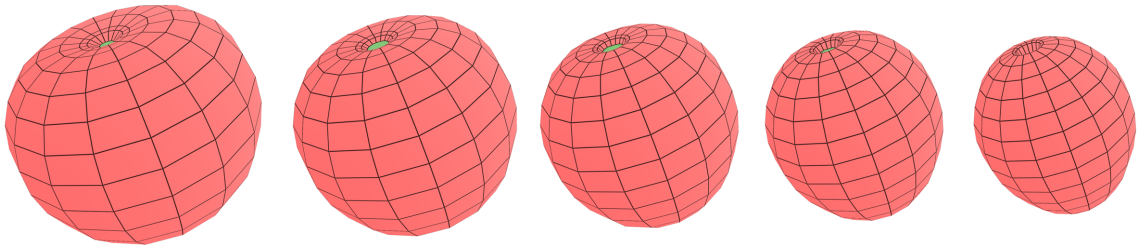


FIGURE 3.30 – Affaissement régulier d'une tomate.

De plus, l'introduction de la règle de pourrissement ciblé, nous a également permis de reproduire le phénomène d'accélération du rétrécissement qu'il est possible d'observer dans les zones où l'intégrité du fruit est compromise. Pour ce faire, la règle de pourrissement ciblé nous a permis d'appliquer un masque de convolution, dont les coefficients sont multipliés par les paramètres K des volumes appartenant au voisinage d'un point spécifique de la surface du fruit, augmentant ainsi localement la vitesse de rétrécissement des volumes. Sur la tomate de a figure 3.31 par exemple, nous avons sélectionné trois points de sa surface, partagés par les volumes de couleur bleu, comme étant le centre de zones où l'intégrité du fruit a été compromise. Nous pouvons voir qu'au fil des applications de la règle, les volumes de la structure arborescente diminuent tous de manière irrégulière.

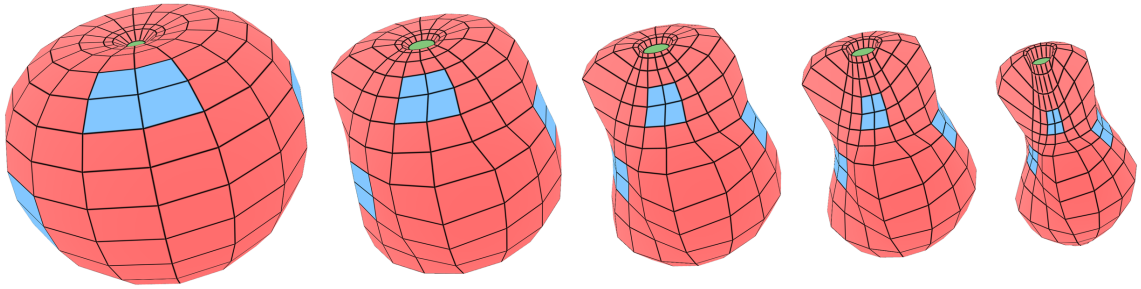


FIGURE 3.31 – Utilisation de la règle de pourrissement ciblé pour accélérer l'affaissement du fruit dans les zones de couleur bleu.

3.4.5 Synthèse

Dans cette section, nous avons introduit un ensemble d'extensions au concept de 3G-cartes L-systèmes qui donnent la possibilité à l'utilisateur de générer des variations géométriques locales à la surface de nos modèles 3D. En nous basant sur les normes définies dans [UNE12] et [OEC02], nous avons pu identifier les excroissances, les crevasses, les impacts de grêle et l'affaissement local comme étant les quatre défauts géométriques locaux les plus souvent rencontrés à la surface des tomates.

Afin de pouvoir cibler des parties spécifiques de la surface d'un fruit dans le but d'y générer des variations, nous avons étendu dans un premier temps le concept de hiérarchie des volumes des 3G-cartes L-systèmes. Grâce aux concepts d'axes, d'étages au sein d'un axe et de chemins, chaque volume de la structure arborescente peut facilement être sélectionné, indépendamment de son label, augmentant alors la puissance du processus de modélisation. Ainsi, en utilisant correctement ce nouveau concept, nous avons pu générer des tomates dont la surface présentait des excroissances de types mucrons ou appendices.

Les crevasses représentaient le deuxième type de variations géométriques locales que nous voulions représenter à la surface de nos fruits. Pour ce faire, nous avons introduit une opération de rupture du bord permettant de rompre les liens qui existent entre deux volumes afin de les séparer et de créer une crevasse au niveau de leur jonction. En nous basant sur [OEC02], nous avons pu identifier les cinq caractéristiques qui définissent une crevasse : sa direction, son épaisseur, sa longueur, sa profondeur et son niveau de cicatrisation. Ainsi, en injectant ces paramètres dans notre opération de rupture du bord, nous avons pu représenter les trois types de crevasses décrites dans [OEC02] : les crevasses longitudinales, concentriques et transversales.

Dans la troisième partie de cette section, nous avons abordé la représentation des impacts de grêle à la surface de nos fruits. Pour ce faire, nous avons introduit deux règles de production. La première, appelée règle d'impact ciblé donne la possibilité à l'utilisateur de sélectionner un endroit spécifique de la surface dans le but d'y provoquer un impact, creusant ainsi un trou de profondeur définie par les paramètres de la règle. La seconde règle quant à elle est appelée règle de génération aléatoire d'impacts, et permet de simuler le côté aléatoire d'une chute de grêle en provoquant un nombre d'impacts défini par l'utilisateur à des endroits définis aléatoirement sur la surface du fruit. Comme nous l'avons vu sur la figure 3.25, l'utilisation de cette règle nous a ainsi permis de générer des dégâts aléatoires à la surface de nos tomates.

Dans la dernière partie de cette section, nous nous sommes enfin attardés sur la simulation du processus d'affaissement du fruit. Pour ce faire nous avons introduit dans un premier temps une règle de rétrécissement permettant simuler la perte d'eau des différents tissus qui forment le fruit. Ensuite, afin d'augmenter le contrôle sur le processus de vieillissement, nous avons introduit un nouveau paramètre de volume, appelé taux de transpiration, qui nous a permis de faire varier la vitesse de rétrécissement des différents volumes en fonction des couches qu'ils représentent. Enfin, dans le but de pouvoir simuler le côté irrégulier de l'affaissement que nous pouvons observer sur les vrais fruits, nous avons introduit un moyen de perturber le rétrécissement d'un modèle 3D en définissant des zones d'accélération du rétrécissement, permettant ainsi d'ajouter du réalisme à l'affaissement de nos fruits 3D.

3.5 Conclusion

Tout au long du cycle de vie d'un fruit, un grand nombre de facteurs différents peuvent compromettre son intégrité. Allant de sa chute à l'attaque de bactéries, en passant par des conditions climatiques défavorables, il est souvent difficile de déterminer la forme qu'aura le produit final. Ainsi, en partant du constat qu'il serait difficile de représenter toutes les gammes de défauts que nous pouvons observer chez toutes les espèces de fruits présents dans la nature, nous nous sommes focalisés dans ce chapitre sur la représentation des défauts géométriques de la tomate. Ainsi, dans la première partie de ce chapitre, nous avons proposé dans un premier temps une méthode permettant de perturber la forme finale des fruits obtenus à l'aide de notre système paramétrique, sans avoir à modifier les valeurs par défaut des variables et des fonctions du système. Cette méthode, basée sur l'attribution aléatoire de valeurs aux différents paramètres du système, couplée à l'utilisation d'une surface englobante permettant de contraindre la forme finale du fruit, nous a ainsi permis de générer des tomates mal formées de formes différentes, et respectant des proportions définies par la surface englobante, à chaque nouvelle exécution de notre système.

À partir du moment où nous avons réussi à représenter des malformations de la forme générale d'une tomate, nous nous sommes attardés dans la seconde partie de ce chapitre à la représentation de variations géométriques locales. Ainsi, en étudiant les normes définies dans [OEC02] et [UNE12], nous avons pu identifier quatre catégories de défauts géométriques locaux que nous avons tenté de représenter dans nos travaux : les excroissances, les crevasses, les impacts de grêle et le phénomène d'affaissement. Dans un premier temps, pour pouvoir cibler des parties spécifiques de la surface de nos fruits, nous avons étendu le concept de hiérarchie des volumes au sein des 3G-cartes L-systèmes en y intégrant la notion de chemins. Ainsi, grâce à ces derniers, nous avons réussi à faire croître des structures de volumes sur des parties spécifiques de nos fruits afin de représenter des excroissances. Ensuite, pour pouvoir générer des crevasses, nous nous sommes basés sur les normes définies dans [OEC02] pour introduire une règle de rupture des bords, permettant de supprimer les relations d'adjacences entre des volumes de la surface. Nous avons ainsi réussi, grâce à cette nouvelle règle, à générer les trois principaux types de crevasses (longitudinales, transversales et concentriques). La règle d'impact aléatoire nous a permis quant à elle de reproduire le côté aléatoire de la chute de grêle en générant aléatoirement des impacts le long de la surface du fruit. Enfin, dans la dernière partie de ce chapitre, nous avons introduit une méthode permettant de simuler la perte d'eau de la tomate, tout en donnant la possibilité d'accélérer localement ce processus.

Conclusion et perspectives

Au cours de cette thèse, nous nous sommes employés à fournir un certain nombre d'outils pour la génération de fruits, de leur structure interne, et de leurs différents défauts de forme. Grâce à des paramètres contrôlant le processus de modélisation, le modèle que nous proposons donne à l'utilisateur la possibilité de créer plusieurs espèces différentes de fruits en grande quantité, et présentant un grand nombre de défauts possibles.

Travaux réalisés

Génération de fruits et de leur structure interne :

Nous avons développé tout d'abord un générateur de fruits basé sur l'utilisation de 3G-cartes L-systèmes paramétriques. Cette catégorie de L-systèmes, basée sur l'utilisation des cartes généralisées de dimension trois, repose sur une subdivision topologique de l'espace en trois dimensions. Ainsi, en partant du constat qu'une grande majorité des fruits présents dans la nature ont tous une structure similaire basée sur une symétrie rotationnelle des carpelles autour d'un axe, nous avons proposé un système simple capable de générer la forme du fruit à l'aide d'une structure arborescente composée d'un axe central autour duquel des axes de premier ordre sont collés pour former le péricarpe.

Par la suite, nous avons cherché à généraliser la forme du fruit dans le but de pouvoir y appliquer des modifications de manière relativement simple. Pour ce faire, nous avons décomposé sa forme relativement complexe en plusieurs petites formes simples définissant chacune une caractéristique bien précise du fruit. La combinaison de ces formes définies par des fonctions mathématiques nous a ainsi permis de créer une version paramétrique de notre 3G-carte L-système donnant la possibilité à l'utilisateur de faire varier la forme globale du fruit de manière intuitive.

Une fois que notre modèle nous a permis de modéliser plusieurs formes de fruits différentes à l'aide d'une seule grammaire, nous y avons introduit une extension donnant à l'utilisateur la possibilité de générer leur structure interne. Pour ce faire, nous avons introduit l'utilisation de la subdivision de Catmull-Clark au travers de trois nouvelles règles de production : une règle de subdivision surfacique, une règle de subdivision volumique et une règle de différenciation. Grâce à ces extensions, nous avons ainsi réussi à subdiviser la structure interne de nos modèles 3D sans modifier leur forme préalablement générée par notre système paramétrique et représenter ainsi les graines ainsi que les trois couches du péricarpe : l'endocarpe, le mésocarpe et l'exocarpe.

Génération de défauts géométriques de tomates :

La deuxième partie de nos travaux porte sur la représentation de défauts géométriques de fruits, en se focalisant tout particulièrement sur le cas de la tomate. Pour ce faire nous avons introduit dans un premier temps une méthode permettant de faire varier la forme globale du fruit de manière aléatoire, tout en la contraignant à respecter des proportions définies par une surface englobante.

Dans la seconde partie de ce chapitre, nous avons ensuite introduit plusieurs outils permettant de générer des anomalies géométriques locales tout autour du fruit. Pour ce faire, nous nous sommes focalisés sur la représentation de différentes catégories de défauts. Dans un premier temps, nous avons ainsi étendu le concept de hiérarchie des volumes en y ajoutant la notion de chemin pour pouvoir cibler des volumes spécifiques de la surface afin d'y créer des excroissances. Dans un second temps, nous avons introduit deux nouvelles règles de production, une règle de rupture du bord et une règle d'impact, qui nous ont permis de représenter respectivement des crevasses et des impacts de grêle à la surface de nos fruits. Enfin, dans la dernière partie de ce mémoire, nous avons proposé une technique donnant la possibilité à l'utilisateur de simuler le processus de rétrécissement du fruit du à la transpiration, tout en lui permettant d'agir localement sur la vitesse à laquelle le fruit perd son eau.

Perspectives

Plusieurs pistes sont envisageables pour la suite de nos travaux. En effet, les travaux qui ont été effectués au cours de cette thèse ne constituent qu'un premier pas dans la création d'un modèle complet de représentation de fruits, de leur structure interne et de leurs défauts de croissance.

Tout d'abord, par rapport à la génération de fruits ayant tous des formes variées, notre système paramétrique ne se limite pour le moment qu'à la génération de fruits appartenant à la catégorie des fruits simples charnus. En effet, comme nous l'avons vu au début de ce mémoire, cette catégorie de fruits représente une grande majorité des espèces présentes dans la nature. Cependant, afin de pouvoir mettre au point un générateur de fruits complet, il serait intéressant de pouvoir représenter les autres catégories existantes. Pour certaines d'entre elles, comme les fruits multiples et les fruits agrégés par exemple, étant donné qu'elles sont composées d'un agglomérat de fruits simples charnus, il ne sera pas nécessaire d'effectuer énormément de changements sur la version actuelle de notre système pour pouvoir les représenter. Des difficultés se présenteront cependant lorsque nous essayerons de représenter des fruits appartenant à la catégorie des fruits complexes. En effet, contrairement aux fruits simples, les fruits de cette catégorie sont issus de la transformation d'autres organes de la plante en plus des ovaires, et leur structure ne présente pas forcément de symétrie rotationnelle autour d'un axe rendant ainsi impossible leur modélisation à l'aide de notre système. Pour répondre à cette problématique, une approche consisterait à chercher des similarités chez tous les fruits de cette catégorie dans le but de pouvoir proposer, à l'instar de notre système paramétrique, un nouveau système spécialisé cette fois-ci dans la représentation de fruits complexes.

En ce qui concerne les défauts de fruits, comme nous l'avons expliqué dans le troisième chapitre de ce mémoire, chaque espèce de fruits présente une très grande variété de défauts qui lui sont propres. Ainsi, la création d'un modèle permettant de représenter tous les types de défauts possibles pour toutes les espèces de fruits existantes serait une tâche extrêmement complexe à effectuer. Cependant, si nous nous focalisons sur le cas de la tomate, certaines pistes peuvent encore être explorées dans le but d'enrichir notre modèle et augmenter ainsi le nombre de défauts qu'il est possible de représenter ou d'améliorer la représentation des défauts déjà existants. Dans le cas des crevasses par exemple, l'opération de rupture du bord se limite à la formation de crevasses le long des arêtes qui sont partagées par des volumes. Une extension intéressante consisterait alors à introduire dans notre modèle la possibilité de créer des crevasses à n'importe quel endroit de la surface du fruit, indépendamment de la topologie du modèle. Une telle opération demanderait de mettre en place une méthode permettant de réadapter la topologie de la structure arborescente en fonction de l'endroit où celle-ci aura lieu, par exemple à l'aide d'un raffinement local.

Au cours de cette thèse, nous nous sommes focalisés sur la représentation de défauts qui sont principalement représentés par l'ajout ou la modification de géométrie sur nos modèles 3D. Ainsi, pour la suite de nos travaux, nous envisageons d'ajouter un moteur de rendu réaliste basé sur l'utilisation de techniques de rendu sous-surfacique spécialement conçues pour les fruits. En plus d'ajouter du réalisme aux modèles 3D générés par notre système, une telle extension nous permettrait ainsi de représenter des défauts de fruits spécialement liés aux matériaux, comme la pourriture par exemple. De plus, l'utilisation d'un moteur de rendu volumique nous donnerait également la possibilité d'augmenter le réalisme des défauts générés à l'aide de notre méthode, comme par exemple l'ajout d'un matériau réaliste aux crevasses ou aux impacts de grêle.

Enfin, dans le but de pouvoir simuler le vieillissement du fruit de manière plus poussée, nous envisageons également d'introduire une notion de temps au sein de notre modèle. Cet ajout, couplé à des méthodes de simulation réaliste, nous permettra ainsi de faire de l'animation pour pouvoir simuler des phénomènes naturels tel que la propagation de la pourriture au sein du volume, l'affaissement dynamique du fruit ou même la croissance de ce dernier.

Bibliographie

[BCS03] Bedrich Benes, Javier Abdul Cordóba, and Juan Miguel Soto. Modeling virtual gardens by autonomous procedural agents. In *TPCG*, pages 58–65. IEEE Computer Society, 2003.

[BCS+11] Richard Bézin, Benoît Crespin, Xavier Skapin, Olivier Terraz, and Philippe Meseure. Topological Operations for Geomorphological Evolution. In *Eurographics workshop on Virtual Reality Interactions and Physical Simulation*, 2011.

[BH94] Sandra Bastin and Kim Henken. Water content of fruits and vegetables. http ://www2.ca.uky.edu/enri/pubs/enri129.pdf. *AGRICULTURE AND NATURAL RESSOURCES*, 1994.

[Bla09] D. Blancard. *Les maladies de la tomate : identifier, connaître, maîtriser*. Ed. Quae, 2009.

[Bli78] James F. Blinn. Simulation of wrinkled surfaces. *SIGGRAPH Comput. Graph.*, 12(3) :286–292, August 1978.

[BSMM11] Bedrich Benes, Ondrej Stava, Radomír Mech, and G. Miller. Guided procedural modeling. *Computer Graphics Forum*, 30, 2011.

[BSW13] Fan Bao, Michael Schwarz, and Peter Wonka. Procedural facade variations from a single layout. *ACM Trans. Graph.*, 32 :8, 2013.

[BTG15] Evans Bohl, Olivier Terraz, and Djamchid Ghazanfarpour. Modeling fruits and their internal structure using parametric 3gmap l-systems. *The Visual Computer*, 31(6-8) :819–829, 2015.

[CC98] E. Catmull and J. Clark. *Seminal Graphics*. ACM, 1998.

[Cho57] Noam Chomsky. *Syntactic Structures*. Mouton, 1957.

[Com15] The Qt Company. Qregexp class | qt 5.5 - qt documentation. http://doc.qt.io/qt-5/qregexp.html, 2015. Accessed : 2015-08-16.

[EMP+02] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling : A Procedural Approach*. Morgan Kaufmann Publishers Inc., 3rd edition, 2002.

[Fri06] Jeffrey Friedl. *Mastering Regular Expressions*. O'Reilly Media, Inc., 2006.

[GBB+07] M Génard, N Bertin, C Borel, P Bussières, H Gautier, R Habib, M Léchaudel, A Lecomte, F Lescourret, P Lobit, and B Quilot. Towards a virtual fruit focusing on quality : modelling features and potential uses. *Journal of experimental botany*, 58 :917–928, 2007.

[GTR+06] Jinwei Gu, Chien-I Tu, Ravi Ramamoorthi, Peter Belhumeur, Wojciech Matusik, and Shree Nayar. Time-varying surface appearance : acquisition, modeling and rendering. *ACM SIGGRAPH 2006 Papers*, pages 762–771, 2006.

[HJT+13] Chun-Yen Huang, Wan-Ting Jheng, Wen-Kai Tai, Chin-Chen Chang, and Der-Lor Way. Special section on procedural modeling : Procedural grape bunch modeling. *Comput. Graph.*, 37 :225–237, 2013.

[KRB11] Joseph T. Kider, Samantha Raja, and Norman I. Badler. Fruit senescence and decay simulation. *Computer Graphics Forum*, 30 :257–266, 2011.

[LCW+12] Youquan Liu, Yanyun Chen, Wen Wu, Nelson Max, and Enhua Wu. Physically based object withering simulation. *Computer Animation and Virtual Worlds*, 23(3-4) :395–406, 2012.

[LD99] Bernd Lintermann and Oliver Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19 :56–65, 1999.

[Lie91] Pascal Lienhardt. Topological models for boundary representation : A comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23 :59–82, 1991.

[Lie94] Pascal Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Int. J. Comput. Geometry Appl.*, 4 :275–324, 1994.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interaction in development : Parts i and ii. *Journal of Theoretical Biology*, 18, 1968.

[LR78] Aristid Lindenmayer and Grzegorz Rozenberg. Parallel generation of maps : Developmental systems for cell layers. In *Graph-Grammars and Their Application to Computer Science and Biology*, volume 73, pages 301–316, 1978.

[LTBG09] Cyril Lam, Olivier Terraz, Nadir Benmounah, and Djamchid Ghazanfarpour. Geometrical aspects of wood aging based on 3gmap l-system growth model. In *Conference - IADIS International Conference Computer Graphics, Visualization, Computer Vision and Image*, 2009.

[LYC+15] Yin Liu, Xiaosong Yang, Yang Cao, Zhao Wang, Biaosong Chen, Jianjun Zhang, and Hongwu Zhang. Dehydration of core/shell fruits. *Computers and Graphics*, 47 :68 – 77, 2015.

[Mau14] James D Mauseth. *Botany : an introduction to plant biology*. Jones & Bartlett Publishers, 2014.

[MP96] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 397–410. ACM, 1996.

[MWH+06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc J. Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25 :614–623, 2006.

[NP15] Radoslaw Niewiadomski and Catherine Pelachaud. The effect of wrinkles, presentation mode, and intensity on the perception of facial actions and full-face expressions of laughter. *ACM Trans. Appl. Percept.*, 12(1) :2 :1–2 :21, 2015.

[OBW+08] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves : A vector representation for smooth-shaded images. *ACM Trans. Graph.*, 27(3) :92 :1–92 :8, 2008.

[OEC02] OECD. *Normalisation internationale des fruits et légumes : la tomate*, 2002.

[ONOI04] Shigeru Owada, Frank Nielsen, Makoto Okabe, and Takeo Igarashi. Volumetric illustration : designing 3d models with internal textures. *ACM Trans. Graph.*, 23 :322–328, 2004.

[PFH00] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 465–470. ACM Press/Addison-Wesley Publishing Co., 2000.

[PHM99] Przemyslaw Prusinkiewicz, P. Hanan, and R. Mech. An l-system-based plant modelling language. *International Workshop AGTIVE'99*, pages 395–410, 1999.

[PL96] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer, 1996.

[PM01] Yoav I. H. Parish and Pascal MÃijller. Procedural modeling of cities. In *SIGGRAPH*, pages 301–308, 2001.

[POB+07] Nico Pietroni, Miguel A. Otaduy, Bernd Bickel, Fabio Ganovelli, and Markus Gross. Texturing internal surfaces from a few cross sections. *Comput. Graph. Forum*, 26 :637–644, 2007.

[Pru04] Przemyslaw Prusinkiewicz. Art and science for life : Designing and growing virtual plants with l-system. *Acta Horticulturae*, 630 :15–28, 2004.

[PTMG08] Alexandre Peyrat, Olivier Terraz, Stephane Merillou, and Eric Galin. Generating vast varieties of realistic leaves with parametric 2gmap l-systems. *Visual Computer*, 24 :807–816, 2008.

[PTSG11] Olga Petrenko, Olivier Terraz, Mateu Sbert, and Djamchid Ghazanfarpour. Interactive flower modeling with 3gmap l-systems. In *Proceedings of 21st International Conference on Computer Graphics and Vision*, pages 20–24, 2011.

[RFL+05] Adam Runions, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-GaÃńlle Rolland-Lagan, and Przemyslaw Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph.*, 24 :702–711, 2005.

[RLP07] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *NPH*, pages 63–70. Eurographics Association, 2007.

[TGM+09] Olivier Terraz, Guillaume Guimberteau, Stephane Merillou, Dimitri Plemenos, and Djamchid Ghazanfarpour. 3gmap l-systems : An application to the modelling of wood. *Visual Computer*, 25 :165–180, 2009.

[TOII08] Kenshi Takayama, Makoto Okabe, Takashi Ijiri, and Takeo Igarashi. Lapped solid textures : filling a model with anisotropic textures. *ACM Trans. Graph.*, 27, 2008.

[TSNI10] Kenshi Takayama, Olga Sorkine, Andrew Nealen, and Takeo Igarashi. Volumetric modeling with diffusion surfaces. *ACM Trans. Graph.*, 29 :180, 2010.

[UNE12] UNECE. *STANDARD FFV-36 concerning the marketing and commercial quality control of tomatoes*, 2012.

[VNM15] Adrien Verhulst, Jean-Marie Normand, and Guillaume Moreau. Visually-realistic appearance changes of fruits and vegetables using particle systems. In *Proceedings of the 32nd Annual Conference of Computer Graphics International*, 2015.

[WKMMT99] Yin Wu, Prem Kalra, Laurent Moccozet, and Nadia Magnenat-Thalmann. Simulating wrinkles and skin aging. *The Visual Computer*, 15(4) :183–198, 1999.

[WYZG11] Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. Multiscale vector volumes. *ACM Trans. Graph.*, 30 :167, 2011.