THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*
Cotutelle internationale *avec l'Université d'Antananarivo*

Présentée et soutenue le *24/05/2022* **par :**

Tiavina Tantely NIVOLALA

Génération automatique d'environnements virtuels urbains complexes

JURY

Gilles GESQUIERE	Rapporteur
Stéphane MERILLOU	Rapporteur
Eric GALIN	Rapporteur
Princy RANDRIAMBOLO-LONDRANTOMALALA	Co-directeur de thèse
Jean-Pierre JESSEL	Directeur de thèse
Véronique GAILDRAT	Présidente
Cédric SANZA	Examinateur
Nancy RODRIGUEZ-DESTRUEL	Examinatrice

École doctorale et spécialité :
MITT : Image, Information, Hypermédia
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse
Directeur(s) de Thèse :
Jean-Pierre JESSEL et *Princy RANDRIAMBOLOLONDRANTOMALALA*
Rapporteurs :
Gilles GESQUIERE, *Stéphane MERILLOU* et *Eric GALIN*

Remerciements

Mes premiers remerciements vont à mes directeurs de thèse : le Pr Jean-Pierre Jessel et le Pr Princy Randriambololondrantomalala, pour m'avoir donné l'opportunité de réaliser cette thèse, pour avoir été à l'écoute, pour leur soutien et encouragements tant techniques que moraux tout au long de la recherche et de la rédaction de la thèse.

J'exprime également ma gratitude au Pr Véronique Gaildrat et au Dr Cédric Sanza de l'IRIT, pou leur acceuil chalereux, les conseils scientifiques, les corrections qui m'ont mis sur la bonne voie et tout leur soutien.

Je remercie le Pr Gilles Gesquière, Pr Stéphane Mérillou et le Pr Eric Galin pour avoir accepté d'être rapporteurs de ce travail, ainsi que Dr Nancy Rodriguez-Destruel pour avoir accepté de participer à ce jury.

Un grand merci aux membres du laboratoire de l'IRIT à Toulouse, ainsi q'aux collègues et amis de l'Université d'Antananarivo, du Laboratoire d'Algèbre et de Géométrie à Antananarivo et de la MISA. A mes amis de Madagascar pour leurs encouragements et leur sollicitude.

Enfin et surtout, merci à ma famille. Mes parents pour leur amour, leur patience, leur soutien tout le long de cette thèse, et sans qui mes études n'auraient pas été réalisables.

Ces travaux ont été menés avec le soutien du Gouvernement Français dans le cadre du programme Territoire d'Innovation, une action du Grand Plan d'Investissement adossé à la 3eme vague du Programme d'investissement d'Avenir (PIA 3), de Toulouse Métropole et du GIS neOCampus de l'Université Toulouse III Paul Sabatier.

Résumé

L'essor des technologies numériques constitue pour les entreprises de la construction, bâtiment et travaux publics (BTP), un enjeu majeur. La maquette numérique, par exemple, procure à tous les acteurs d'un chantier une visibilité en temps réel des différents aspects mis en jeux. Dans de nombreux domaines, il est donc nécessaire de modéliser en 3D des immeubles, des bâtiments industriels, des ensembles immobiliers, des villes ... Le BIM (Building information modeling, ou building information model, en français modélisation des données du bâtiment) est un ensemble de technologies et de processus d'intégration, de production, de gestion et de visualisation de données des modèles de construction du BTP.

Plusieurs voies existent pour éviter aux concepteurs en architecture, aux urbanistes ou aux graphistes de fastidieuses opérations manuelles pour la création de ces environnements, des techniques de génération automatique ont été développées en complément (ou en substitution) des logiciels de modélisation et de conception assistée par ordinateur (CAO) qu'ils utilisent pour créer ces environnement virtuels. De nombreux systèmes procèdent à la génération de ces environnements 3D à partir du monde réel : les scanners 3D ou les prises de vue stéréoscopiques permettent en effet de reconstruire des objets, des bâtiments et des environnements 3D à partit des données géométriques et photométriques acquises.

Dans un autre approche, les techniques de modélisation procédurales permettent de générer automatiquement des environnements 3D complexes en utilisant des déclarations de propriétés ou de contraintes issues de règles métier et des techniques issues de la Vie Artificielle. Les données issues de ces systèmes d'acquisition ou de génération peuvent être enregistrés et gérées dans les systèmes d'informations géographiques et les logiciels de cartogra-

phie ou dans une suite logicielle BIM. Les données de ces applications informatiques et les métadonnées associées qu'ils contiennent sont également exploitables pour générer ou modifier les environnements 3D.

Parmi toutes les méthodes procédurales qui ont été utilisées pour aborder la question, les grammaires de formes sont particulièrement efficaces face au problème d'encodage des connaissances sur les formes et leur reproduction. Le but de la thèse est de générer des environnements urbains en 3D incluant des données issues de l'existant et de nouveaux édifices à travers l'utilisation des grammaires de formes. En partant d'une interface graphique, de données géographiques et de descripteurs, le but est de générer les règles encodant les informations d'apparence des bâtiments de manière semi-automatique. A l'issue de ce processus, un fichier BIM de ces bâtiments au format standard IFC pourra être généré.

Table des matières

Table des figures

Chapitre 1

Introduction

Sommaire

1.1 Contexte et objectifs

Les agglomérations urbaines sont des structures excessivement complexes, parfois très étendues et reliées à une quantité massive de données à définir et à traiter. L'intérêt pour l'étude des environnements urbains s'est considérablement accru avec les récents développements des technologies de l'information, la modélisation géométrique et la visualisation. Ces environnements urbains sont composés d'objets et peuplés d'entités, tous créés par l'homme. Ces espaces nécessitent le stockage de collections complexes, correspondant à différents niveaux d'informations, chacun correspondant à certains types d'éléments : la modélisation du paysage, les réseaux hydriques, des réseaux routiers, l'architecture des bâtiments. Chaque élément joue son propre rôle dans la définition d'un tel ensemble d'informations, et chaque ville possède une structure unique qui lui est propre, définissant sa propre dynamique. Chacun des éléments permettant la description d'un environnement urbain possède une caractéristique intéressante, ils peuvent être divisés de manière cohérente en structures régulières à partir desquelles il est possible d'extraire

les hiérarchies sous-jacentes.

On constate que de telles structures sont principalement composées de parties décrites par des motifs géométriques observables. Afin de recréer des formes similaires, la connaissance de ces structures devient essentielle.

Les différents styles architecturaux que l'on peut observer dans différents milieux urbains peuvent être décrits par la façon dont leurs éléments sont composés, qui à leur tour peuvent être définis par un ensemble de règles. Les grammaires de formes ont été utilisées avec succès pour la construction et l'analyse des architectures, mais également pour modéliser la végétation, les intérieurs des bâtiments, les rues, les sculptures, les meubles, etc. Les grammaires sont connues pour leur efficacité et la facilité de prise en main de leurs méthodes de création et de modélisation de paysages urbains. Elles fournissent un formalisme permettant de créer de nouveaux modèles, mais sont également utiles dans l'analyse d'un style par décomposition. En effet, les grammaires permettent de définir et d'organiser les critères nécessaires à la caractérisation d'une architecture, et permettent ainsi de déterminer si une conception appartient à un certain style ou non. Elles fournissent également les composants, nécessaires pour définir de nouveaux styles et d'en créer de nouvelles instances. Cela fait de l'utilisation des grammaires en général, et des grammaires de forme en particulier, une technique appropriée à la représentation des connaissances sur l'architecture, afin de permettre la génération d'une grande variété de modèles similaires mais pour autant non identiques. Étant donné que la création des ensembles de règles nécessaires est une tâche non triviale, il est primordial de faciliter à l'utilisateur, non seulement la fourniture des directives essentielles mais aussi des informations supplémentaires, afin de capturer au mieux une partie de ses connaissances, de façon à rester fidèle à ce qu'il souhaite recréer.

La disponibilité des données permet désormais d'intégrer des géométries complexes dans un système standard, et bien qu'en ce moment, il n'existe pas encore de standard unique dans ce domaine. Quelques modèles se distinguent par leur qualité et la complexité qu'ils permettent d'obtenir.

Les développements, notamment dans le domaine des systèmes d'informations géographiques, ont permis de répondre aux demandes de création de logiciels pour interroger les structures de données spatiales et visualiser les

résultats de ces requêtes sous forme de modèles 3D. Différents efforts de création se sont accrus, en conjonction avec l'émergence de diverses techniques de modélisation et la multitude d'utilisations. Tout cela a contribué à rendre la modélisation et la visualisation 3D d'environnements urbains plus répandus et surtout accessibles.

Au cours de ces dernières années, les systèmes d'informations géographiques se sont étoffés et complexifiés pour prendre en compte les informations relatives aux objets 3D. Les utilisations les plus courantes des modèles de villes 3D se situent dans les domaines de l'urbanisme, de l'architecture et de l'aménagement du paysage. Ces modèles permettent aux concepteurs, et aux architectes de parcourir virtuellement un projet pour obtenir une perspective plus immersive de leur travail. Il existe, pour ces représentations, une variété d'approches utilisant différentes sources de données et visant des objectifs pouvant être spécifiques. Le choix de la méthode doit prendre en considération le type de projet, les caractéristiques des objets à représenter, les sources de données disponibles, ainsi que les contraintes logicielles et matérielles.

Dans ce cadre, il apparaît que le potentiel de la création procédurale ne peut être ignoré car c'est une méthode efficace pour créer des modèles tridimensionnels d'une manière rapide et polyvalente. La modélisation procédurale a déjà été utilisée pour automatiser ou semi-automatiser la création de structures complexes, basées sur un ensemble de règles et de propriétés dans plusieurs travaux antérieurs. Les langages de modélisation utilisant les grammaires sont d'excellents exemples de la façon dont le processus de modélisation d'une zone urbaine peut être réalisé de manière rapide et efficace.

Dans ce contexte, nous nous sommes intéressés au modèle IFC (Industry Foundation Classes) : un format de fichier destiné à décrire les données de l'architecture, du bâtiment et de l'industrie de la construction, fréquemment utilisé dans les projets basés sur la modélisation des informations du bâtiment (BIM : Building Information Modeling), et à la création de règles grammaticales expressives. Or, la complexité de la prise en compte de ces deux aspects a mis en exergue la nécessité de l'introduction de l'automatisation dans ces domaines.

La création d'un modèle BIM a été rarement automatisée ou associée

à la modélisation procédurale. Notre travail s'est focalisé sur les méthodes de création automatique de ces modèles de bâtiments. Le principal objectif de nos travaux est donc la création d'un prototype capable de faciliter la création d'une grammaire et le modèle IFC résultant. Plusieurs sous objectifs en découlent :

— l'étude des représentations des environnements urbains
— la conceptualisation du prototype
— l'évaluation des méthodes et de la qualité des résultats obtenus

Le but de notre projet est ainsi de rechercher et de combiner diverses techniques procédurales, afin de trouver une approche appropriée permettant de simplifier et rendre accessible la génération de modèles de bâtiments 3D pour l'interaction, la visualisation d'environnements urbains complexes. Ce projet doit éventuellement permettre aux concepteurs de raffiner et d'intégrer ces modèles dans d'autres médias. Nos principaux objectifs pouvant être déclinés de la façon suivante :

— obtenir un modèle réaliste du point de vue visuel et métier et avec une grande facilité de réalisation
— offrir différentes façons d'automatiser la création des règles
— et enfin, obtenir un modèle réutilisable et sémantiquement riche

Nos contributions concernent ainsi principalement :

— dans le BIM : la création d'un système faisant le lien entre les concepts hétérogènes de grammaire, les SIG et la représentation IFC
— dans l'automatisation des grammaires : la possibilité de générer les grammaires via une interface et l'automatisation de la recherche d'objectifs esthétiques pour les bâtiments

1.2 Organisation du document

Le document est organisé comme suit :

Chapitre 1 : Introduction.
Cette section présente un aperçu du travail de recherche qui a été mené. Il décrit le contexte, l'énoncé du problème, la problématique soulevée et les objectifs de la recherche.

Chapitre 2 : Etat de l'art.
Ce chapitre traite des travaux antérieurs en se focalisant sur nos principaux objectifs de recherche : les techniques de modélisation de villes virtuelles en 3D et les types de représentations standards, le tout avec une analyse des avantages et des inconvénients des différentes techniques abordées. L'étude des méthodes et des techniques existantes nous aidera ainsi à comparer et à évaluer les différentes solutions et à nous concentrer sur les plus pertinentes pour notre travail.

Chapitre 3 : Contribution : la modélisation des bâtiments.
Ce chapitre se concentre sur l'étude préliminaire et sur le prototype que nous avons créé. Il explique les choix effectués par rapport au modèle et les méthodes choisies et il détaille la conception du traitement (de type pipeline), de la création de la base de connaissance et la création de l'interpréteur de grammaire et le cheminement jusqu'à l'obtention du modèle IFC. Il introduit également l'automatisation de la création des règles grammaticales via l'utilisation d'algorithmes génétiques.

Chapitre 4 : Discussion.
Ce chapitre est une discussion et une analyse des travaux présentés dans les chapitres précédents. Il permet d'évaluer quantitativement et qualitativement les différents résultats et de les comparer avec des données réelles.

Chapitre 5 : Conclusion et perspectives.
Ce dernier chapitre donne une réponse aux questions de recherche soulevés dans ce mémoire par une conclusion générale et ouvre une fenêtre vers l'approfondissement de l'étude et de l'amélioration de notre travail, en justifiant les limites techniques, et en donnant des pistes de réflexions pour une éventuelle résolution.

Chapitre 2

État de l'art et problématique

Sommaire

Dans ce qui suit, nous présenterons différentes approches pour modéliser et de simuler les principaux éléments d'un environnement urbain : les bâtiments et les réseaux routiers, en commençant par ces derniers.

2.1 Génération des rues

Les réseaux routiers sont aussi complexes que les nombreuses formes qu'ils peuvent avoir, de ce fait, ces réseaux routiers peuvent être représentés par plusieurs concepts : intuitivement sous forme de graphe géométrique par exemple. De nombreux modèles dans la nature peuvent également donner lieu à une structure de rues intéréssante : les diagrammes de Voronoi, les arbres binaires, ... Plusieurs méthodes ont été proposées pour dessiner et éditer ces rues parmi lesquelles des approches basées sur des modèles préexistants, les L-systèmes, les approches multi-agents et les champs tensoriels.

La modélisation des rues existantes et la création de nouvelles dans un ensemble urbain se base sur la génération d'un modèle suffisamment plausible, à partir des données disponibles. Celles-ci peuvent être obtenues avec des données SIG généralement via OSM qui est une carte modifiable en ligne gratuite, cela peut ensuite servir d'exemple pour une nouvelle génération. L'influence des cartes de terrain sous-jacentes et des données d'élévation peuvent aussi être pris en compte à des degrés divers.

Aliaga, Vanegas et Benes [1] proposent une méthode de génération à base d'exemples de structures et des images d'aménagements urbains réels, en permettant à l'utilisateur de créer un nouvel environnement urbain grâce à des opérations interactives d'extension, de mélange et de combinaison, sans avoir à se soucier des détails structurels de bas niveau.

Ce système prend en entrée les données vectorielles des rues, des blocs et des parcelles de l'espace urbain, ainsi que des images aériennes, et considère la connectivité et le zonage des parcelles et des rues. L'espace urbain est décomposé en une collection de tuiles adjacentes, séparées par des limites de route ou de parcelle. Le système construit ensuite de manière interactive un nouvel aménagement en créant et en joignant de nouveaux fragments. Plusieurs opérations d'édition, telles que développer, redimensionner, remplacer et déplacer, sont prises en charge.

Un algorithme de synthèse de structure permet de créer les rues, une méthode de synthèse d'images génère des images aériennes en réutilisant des données des exemples en entrée, et un système de synthèse de tracé fournit un ensemble d'outils interactifs pour assembler un nouvel aménagement. Pour générer un nouveau tracé, l'utilisateur copie et place de manière interactive des points d'intersection du réseau routier concerné. Les transformations sont effectués en répartissant la déformation globale résultante entre toutes les tuiles, tout en préservant leur connectivité et en minimisant leur distorsion individuelle.

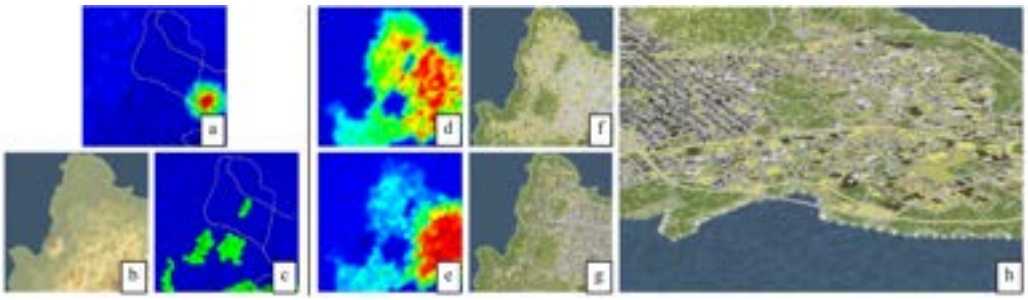

Figure 2.1 – Prenant comme entrée un ensemble de points de repère spécifiés par l'utilisateur (emplacement du centre-ville (a), terrain et autoroutes (b) et parcs (c)), le système complète automatiquement le reste de la ville. Le processus consiste à calculer d'abord les variables comportementales (population (d) et emplois (e)) puis les variables géométriques (blocs, parcelles (f) et bâtiments (g)) dans le respect des repères. Une vue du modèle 3D résultant de la ville est représentée en (h). Source : Vanegas et al. [44]

Vanegas et al. [44] utilisent une approche basée sur la conception interactive de modèles 3D de larges espaces urbains. Leur approche, basée sur la

simulation, est capable de générer un modèle urbain complet avec une spécification partielle seulement par le concepteur. La modélisation géométrique et comportementale est utilisée en conjonction, en se basant notamment sur l'hypothèse que la forme des réseaux routiers est difficile de recréer uniquement à partir de paramètres comportementaux ou géométriques.

Le moteur est composé d'un générateur de réseau routier adaptatif, d'un générateur de parcelles et d'un générateur de bâtiments. Il utilise des variables géométriques (longueur d'une route, volume des bâtiments, ...) et comportementales (population, nombre d'emplois) qui sont liées entre elles. Ces variables peuvent être modifiées par l'utilisateur avec un pinceau. Si une variable est modifiée par l'utilisateur, le système réagit et met à jour automatiquement les autres variables en utilisant des algorithmes adaptatifs pour régénérer la géométrie, et redistribuer la population et les emplois jusqu'à retrouver un état d'équilibre. Les équations différentielles déterminent les changements de variables géométriques par cellule de grille.

Finalement le réseau routier est généré en s'adaptant à ces informations : un ensemble de graines est généré en tenant compte de la répartition de la population et des emplois, chacune est convertie en une intersection du réseau routier et est utilisée pour générer des segments de routes le long desquelles les rues sont générées. Les enveloppes de bâtiment sont générées en estimant leur taille à partir de la population et des emplois qu'elles doivent contenir. Des modèles 3D de complexité intermédiaire sont utilisés pour fournir un retour visuel à l'utilisateur pendant le processus interactif de conception. Le processus est montré étape par étape dans la figure 2.1.

2.1.1 Champs tensoriels

Les travaux de Chen et al. [10] apportent de nombreuses contributions en connectant les graphes routiers aux champs tensoriels. Un champ tensoriel est une fonction continue qui associe tout point p = (x,y) à un tenseur. Un tenseur fait référence à une matrice symétrique de dimension 2x2.

Leur approche interactive se base sur des champs tensoriels crées par l'utilisateur, lui offrant un niveau de flexibilité sans précédent à l'époque en permettant d'éditer ces dernières et spécifier des contraintes telles que des

motifs désirables et des informations topologiques.

Le système accepte quatre cartes en entrée : les zone hydrographiques, les parcs et forêts, les élévations et la densité de la population. Chaque contrainte est convertie en tenseur basique et des outils interactifs permettent à l'utilisateur de manipuler et créer des tenseurs localement à l'aide de la souris. Tout ces champs sont combinés comme montré dans la figure 2.2..

Le graphe est ensuite généré en calculant les hyperstreamlines du champ tensoriel. Un hyperstreamline définit les courbes tangentes à un champ de vecteurs propres le long de leurs trajectoires. Le réseau routier est représenté par les sommets qui sont une collection des points d'intersection entre les hyperstreamlines et les arêtes qui sont l'ensemble des segments entre deux intersections consécutives. Les attributs concernant ces rues peuvent être également stockées dans le graphe. Ce graphe peut être édité après génération en créant, manipulant ou supprimant des segments, vertices, et en insérant ou déplaçant des nouvelles rues, pour toute nouvelle insertion, le réseau existant s'adaptera automatiquement autour. La troisième étape est finalement la génération de géométries 3D sur le terrain résultant, cette génération est gérée par le moteur CityEngine [35].

2.1.2 Système multi-agent

Dans les travaux de Lechner et al. [28], la simulation se base sur les systèmes multi-agents et l'ensemble de règles qui les régit. Leur but étant non pas de reproduire une ville existante mais de générer des villes qui soient réalistes et plausibles. Chaque agent se focalise sur une parcelle de terrain rectangulaire et n'est pas directement concerné par les actions d'autres agents aux alentours, la simplicité de ces règles permet de réduire les conflits et incohérences.

La description du terrain est la seule donnée nécessaire en entrée, l'utilisateur peut également spécifier des attributs et paramètres tels qu'un motif en grille pour le réseau routier ou la distance minimale entre deux rues. Cette création progressive du réseau est montrée dans la figure 2.3.

Le système de rues est généré à l'aide de deux types d'agents : connecteur et extendeur. L'agent connecteur traverse le réseau routier existant et

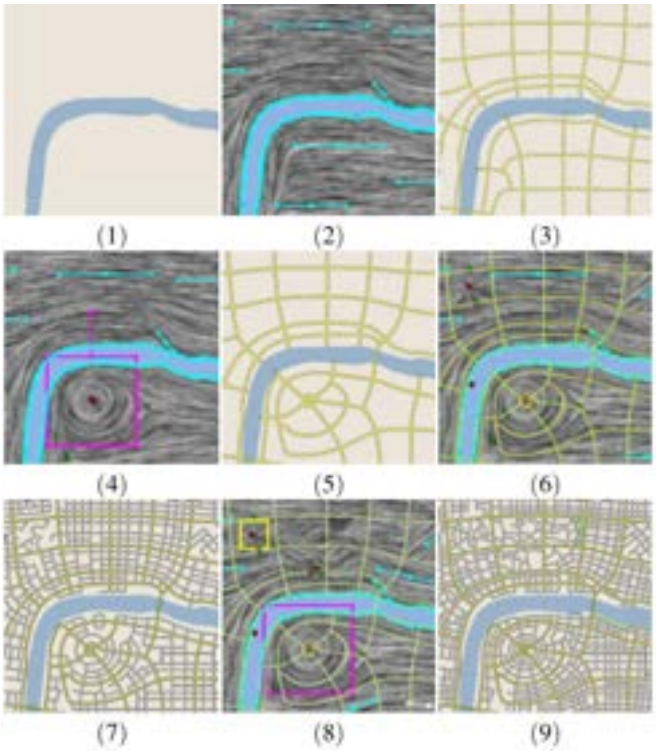

Figure 2.2 – Les étapes de la modélisation. Source : Chen et al. [10]

construit au hasard un segment de rue vers toute parcelle de terrain qu'il ne peut pas atteindre dans un certain rayon prédéfini. L'agent extendeur traverse le terrain et connecte les parcelles non connectées au réseau, des test sont effectués pour s'assurer que le placement du segment est optimal.

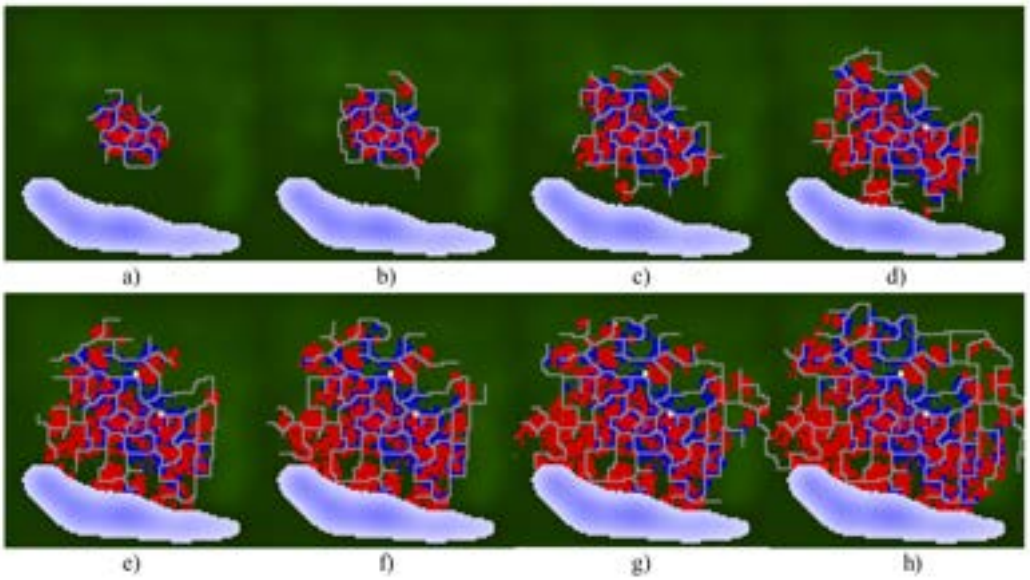

Figure 2.3 – Une progression des résultats à intervalles successifs lors d'une simulation d'une ville organique. Source : Lechner et al. [28]

Un autre type d'agent développe les bâtiments commerciaux et résidentiels en traversant le terrain et suggérant de bâtir sur un morceau de terrain vide si cela augmente la valeur de cette parcelle, et suggère un plan de construction pour ce site.

2.1.3 L-Système

Une des méthodes de génération les plus fréquentes est basée sur l'utilisation des L-système. À l'instar des modèles végétaux, un réseau routier peut être considéré comme une structure en croissance et peut être simulé par un système de réécriture. Parish et Muller utilisent un L-système étendu pour développer un réseau routier [32]. Le L-système est axé autour d'objectifs ; comme la densité de population, et des tracés routiers spécifiques, par exemple le raster ou le motif radial. Ce L-système est étendu avec des règles qui ont tendance à relier les nouvelles routes proposées aux intersections existantes et des règles qui vérifient la validité de la route. Les petites rues sont insérées dans les zones restantes à l'aide d'une grille. Cette méthode est entièrement automatique et fournit un contrôle limité uniquement

en définissant des cartes de densité de population.

Lors de l'écriture d'un système de règles complexe pour créer un plan de rue, un grand nombre de paramètres et de conditions doivent être respectées. Les fonctions validant ces contraintes suivent une hiérarchie pour faire la distinction entre les tâches de niveau supérieur et les contraintes environnementales appelées contraintes globales et contraintes locales respectivement. Lorsqu'elles sont invoquées, les contraintes globales et les contraintes locales prennent en compte plusieurs influences qui peuvent définir ou modifier les valeurs des paramètres. L'outil de création de routes fait la distinction entre les objectifs globaux et les contraintes locales comme suit : les objectifs globaux sont des modèles de rues et la densité de population et les contraintes locales sont les limites du genre terre/eau/parc, élévation et rues. La figure 2.4 montre, à gauche, deux cartes de contrôle qui pondèrent respectivement les modèles de rue de New York et de Paris et à droite le résultat issu de la combinaison de ces deux modèles.

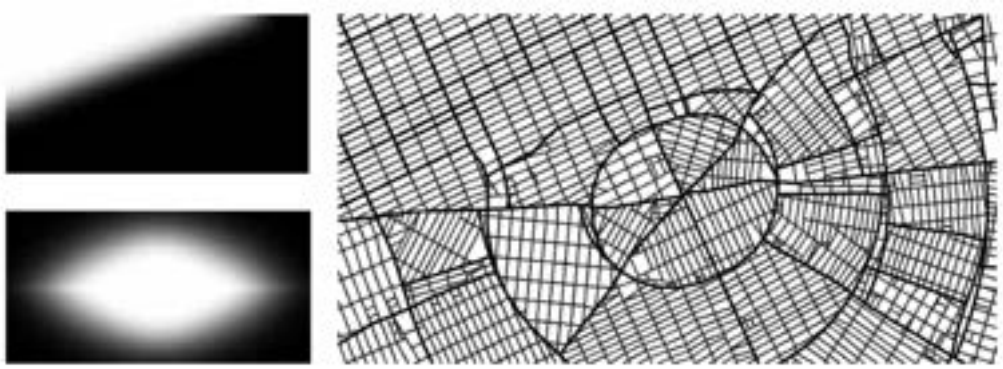

Figure 2.4 – Deux cartes de contrôle de modèles pour la règle de New York et de Paris. Droite : Le motif de rue résultant. Müller et al. [32]

2.1.4 Evolution interactive

Une autre technique [30] utilise des langages formels pour générer un processus croissant partant des axiomes, le réseau routier étant représenté en tant que graphe ou arbre avec différents niveaux de complexité selon les règles. Le système de modélisation est interactif et l'utilisateur est capable de faire évoluer des règles grammaticales. Le système est basé sur un analyseur qui

prend un ensemble de règles et de paramètres et crée un modèle géométrique.

Le génotype parent est un ensemble de règles qui sont mutées selon des probabilités spécifiques et la sélection est basée sur les aspects que l'utilisateur trouve esthétiques. À l'issue du processus de mutation et de génération, l'utilisateur peut manipuler et examiner de manière interactive les phénotypes, et sélectionne quel phénotype est le plus approprié pour devenir le nouveau parent. Le processus de mutation, génération, sélection est répété jusqu'à ce qu'une forme satisfaisante soit atteinte ou au bout d'un temps défini.

2.1.5 Grammaires de graphes

L'approche de Fiser et al. [17], basée sur la modélisation procédurale inverse a pour but de représenter une structure du monde réel, notamment la détection de structures tels que les réseaux routiers et d'éffectuer la synthèse de ces dernières. Leur approche va utiliser les modèles procéduraux combinés aux grammaires de graphes. Ces grammaires de graphes contiennent des informations géométriques, et encodent des données topologiques avec cette géométrie.

Pour définir ce graphe géométrique : chaque sommet a des coordonnées x et y, les règles de production de la grammaire remplacent un nœud par un graphe et son plongement, enfin la réécriture sera toujours autorisée même avec une connectivité partielle entre les nœuds nouvellement insérés et les nœuds existants.

Pour l'apprentissage, l'analyseur graphique va chercher un sous-graphe qui se répète fréquemment et créer des règles de réécriture en remplaçant le sous-graphe par un seul sommet, puis il va chercher à trouver une grammaire qui codifie efficacement les structures répétées d'un graphe et l'analyseur de graphe trouve les modèles les plus fréquemment répétés. En utilisant l'expansion des sommets de groupes de graphes isomorphes, une fois les sous-graphes détectés, ils sont codés dans une grammaire de graphes géométriques. Cette grammaire est utilisée dans leurs travaux pour la création de réseaux routiers en rassemblant plusieurs types de réseau routiers précédemment étudiés.

2.2 Génération des bâtiments

La génération procédurale de bâtiments est l'un des domaines les plus développés. La plupart des méthodes de cette catégorie utilisent une forme de système de réécriture formelle, comme un L-système, ou une grammaire de forme comme base pour générer un modèle de bâtiment 3D à partir d'une empreinte 2D. Ces méthodes peuvent être utilisées pour créer des bâtiments détaillés et convaincants partant de très peu d'informations en entrée, mais nécessitent beaucoup d'efforts de création. Certaines méthodes alternatives tentent de reconstruire automatiquement des grammaires à partir d'ensembles de données du monde réel, comme des photographies de façades de bâtiments, ou encore la télédétection. La visualisation a dominé les premières utilisations des modèles de ville en 3D. Mais au fur et à mesure que la technologie se développait, ces modèles ont été utilisés à plusieurs fins dans un grand nombre de domaines [38] [6] [42].

2.2.1 Génération procédurale

Générer des modèles de manière procédurale s'avère être une technique efficace pour générer des modèles 3D, principalement en améliorant les données existantes.

Ils trouvent leur utilisation dans les SIG dans une gamme croissante d'applications comme la planification urbaine et la simulation [4] [36]. Un exemple de processus consiste à prendre des modèles de blocs de bâtiments et à synthétiquement augmenter leurs détails et leur apparence (l'ajout d'une forme de toit et d'une façade texturée) selon une architecture prédéfinie typique de cette étendue spatiale, il en résulte un modèle de ville en 3D avec un niveau de détail beaucoup plus fin sans coût supplémentaire significatif. Un autre exemple est de prendre des empreintes de bâtiment en entrée, par exemple détectées à partir d'un nuage de points [21], et pour y générer des bâtiments au-dessus. Un avantage de cette technique est la rapidité et la simplicité pour générer des modèles de villes 3D, généralement en grande quantité. De plus, les modèles dérivés de la modélisation procédurale sont expressifs dans les détails et des systèmes de vérifications peuvent être implémentés de manière à ce qu'ils contiennent rarement des incohérences topologiques.

Demir, Aliaga et Benes [15] proposent d'assister les tâches de modélisation en réutilisant des modèles existants tels que les maillages architecturaux, les nuages de points ou les zones urbaines texturées. Pour ce faire, ils utilisent des méthodes d'analyse de forme afin de révéler les informations cachées dans les modèles existants, qu'ils soient géométriques ou sémantiques, notament la répétition et les informations structurelles. Des méthodes de découverte de grammaire sont ensuite utilisées sur ces composants pour en extraire une qui représente le modèle.

Premièrement les outils de traitement de forme vont segmenter et étiqueter les nuages de points de manière semi-automatique en se basant sur la similarité, puis découvrir des composants en regroupant ces composants par un vecteur de caractéristiques

La grammaire peut être découverte de plusieurs façons : soit par un algorithme qui convertit les segments précédemment trouvés en une représentation procédurale, soit avec une approche qui convertit les nuages de points des bâtiments en une représentation procédurale tout en complétant ces modèles et en affinant leur reconstruction, ou encore en trouvant les modèles et la grammaire à partir des composants précédemment découverts dans une ville. Cela va permettre la synthèse de ces modèles, l'interrogation et la simplification sur de grandes aires urbaines.

Une fois l'information structurelle découverte, la configuration spatiale obtenue peut alors être utilisée pour recréer et reconstruire les bâtiments de manière automatique.

Toujours sur les nuages de points, les travaux de Demir, Aliaga et Benes [13] consistent à inspecter ces nuages de points afin de segmenter et d'organiser les structures répétitives en une représentation hiérarchique, et découvrir et organiser ces répétitions en améliorant à la fois l'exhaustivité et la qualité des structures échantillonnées. Ces structures sont ensuite complétées ce qui permet une édition ultérieure. Cela afin d'effectuer directement un ensemble d'opérations sur l'environnement urbain.

La première étape est la segmentation, c'est à dire la découverte des plans dominants à l'aide de la correspondance de modèles, l'utilisateur peut sélectionner de manière interactive une région d'intérêt et générer une seg-

mentation supplémentaire, puis les faces planes sont décomposées en nuages de points répétitifs. Une fois que l'utilisateur identifie un segment de modèle qui se répète, le système balaye le volume sélectionné dans chacune des trois directions axiales par rapport aux coordonnées du segment pour identifier les répétitions.

La deuxième étape convertit le motif découvert en représentation grammaticale, il s'agit de construire une arborescence qui contient tous les segments et leurs répétitions, de manière descendante : les nœuds vont représenter des sous-ensembles du modèle et les arêtes représentent les opérations de découpage. Enfin il y aura l'extraction de motifs : la méthode commence par placer tous les nœuds visibles ayant la même étiquette dans le même groupe. Chaque groupe est inspecté pour vérifier si les sous-arbres de tous les membres du groupe sont similaires. Si ce n'est pas le cas, les membres du groupe sont répartis dans un nouveau groupe. Ces étapes sont répétées jusqu'à ce qu'il n'y ait plus de changement de groupe.

Pour la synthèse, l'utilisateur peut créer de nouveaux nuages de points avec une méthode qui utilise la grammaire découverte. Les nuages de points pourront être édités au moyen d'un ensemble d'opérations d'édition locales et globales, d'optimisation et de ré-échantillonnage.

Cet outil permet de modifier directement la grammaire générée et prend en charge l'édition interactive telles que : redimensionnement intelligent, copier puis insérer une règle ou un terminal existant du modèle avec redimensionnement du contenu final, basé sur l'arbre construit précédemment. Le modèle original peut être modifié pour produire un nouveau modèle de qualité similaire.

Zhou et al. [49] proposent une approche de rendu hybride qui vise à simplifier la visualisation du modèle urbain à moyenne et longue distance, leur stratégie consiste à sélectionner et restituer dynamiquement des points et lignes du rendu en fonction de la superficie raster des objets. L'entrée du système est la géométrie et les textures du bâtiment. Les lignes sont générées en analysant la géométrie du modèle et le regroupement des couleurs basé sur l'image. Les résultats de cette étape sont des séquences progressives de points et de lignes où des représentations simplifiées d'un modèle urbain peuvent être générées.

Pour la génération et le classement du niveau de détail, la distribution des points doit suivre plusieurs caractéristiques importantes. La génération de lignes est contrôlée par l'angle entre deux faces de maillage adjacentes. Si l'angle est inférieur à un certain seuil, un seul bord adjacent sera extrait et représenté sous forme de contours d'une structure principale. Les informations de couleur sont converties en espace colorimétrique dans lequel les différences de couleur mesurées sont proportionnelles à la perception humaine et peuvent être calculées comme la distance euclidienne. Le classement des points crée une séquence de points pour un modèle d'entrée. Cette représentation progressive permet de générer une simplification du modèle de la zone donnée à différents niveaux de détails.

2.2.2 Approche par l'extraction de données terrestres et aéroportées

Pour obtenir des modèles de villes riches en détails, Dold et Brenner [16] utilisent la fusion et la mise en correspondance automatique de scans lasers 3D. L'approche permet d'enregistrer les balayages laser terrestres sans utiliser de points de référence pour déterminer les paramètres de transformation. Un exemple de scan laser de l'opéra de Hannover peut être observé dans la figure 2.5. Les plans sont extraits de ces scans lasers via un algorithme de croissance qui consiste à définir une première région, et à étudier les points voisins pour vérifier qu'ils peuvent être ajoutés à cette région. Les parcelles planaires sont dérivés des positions de balayage uniques des données du scan laser : les points voisins sont lus et peuvent être utilisés pour estimer un plan qui correspond aux points mesurés. Une nouvelle région est ensuite sélectionnée parmi les points restants et le processus se répète. Les données provenant du balayage laser sont stockées dans une image contenant des couches avec les coordonnées nécessaires.

Après avoir extrait des plans des données de balayage, l'étape suivante du processus de mise en correspondance consiste à calculer les paramètres de transformation d'un mouvement rigide entre deux positions de balayage différentes. Les méthodes de Grimson, Huttenlocher et al. [18] et Bunke et Jiang [7] sont utilisées pour la détermination des paramètres de transforma-

Figure 2.5 – Scan laser de l'opéra de Hannover. Source : Dold et Brenner [16]

tion.

2.2.3 Approche par la modélisation procédurale inverse

Pour faire face à l'incertitude par rapport à la qualité des données, Zhang et al. [48] utilisent l'imagerie satellitaire en conjonction avec des données supplémentaires pour produire un modèle de ville sur une zone cible : concrètement le système crée des disposition spatiales des parcelles et des bâtiments avec la modélisation procédurale inverse.

Comme les caractéristiques statistiques d'une ville sont observables dans des images satellites qui se traduisent par des apparences distinctes à grande échelle, un apprentissage Deep Learning sera utilisé pour produire des villes statistiquement similaires de loin : en termes de nombre moyen de bâtiments et la zone de construction et leur distribution. Le système se sert d'images satellites d'entrée qui ont été segmentées et étiquetées, d'OpenStreetMap pour les routes, des données de hauteur mondiale et données de population. Leur méthode déduit un modèle de ville contenant des détails plausibles.

Le processus se déroule en 3 étapes : l'estimation de la surface des parcelles, la génération de modèles procéduraux en utilisant cette classification, enfin les données incertaines sont complétées par les parcelles, les contours des bâtiments et la géométrie.

La relation entre la taille des parcelles et la taille des bâtiments dans deux grandes villes est analysée pour former un réseau de classification, et en utilisant les tailles, les données, la géométrie des bâtiments, la superficie

des parcelles sera estimée pour approximer la disposition des parcelles d'un bloc et une zone urbaine entière est créé.

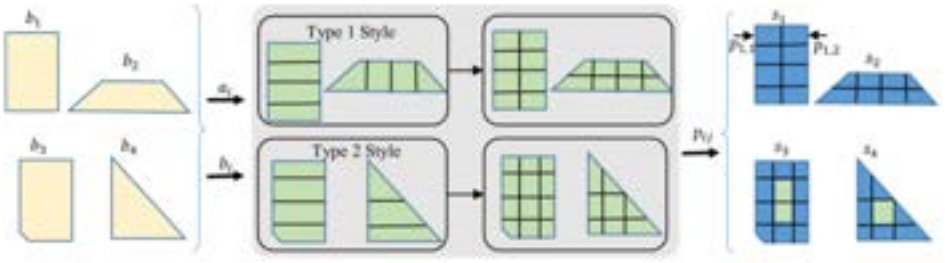

Figure 2.6 – Génération de parcelle. La méthode prend en entrée une image satellite d'un pâté de maisons ainsi que l'estimation superficie de parcelle dans l'image et génère une subdivision parcellaire. Source : Zhang et al. [48]

Demir, Aliaga et Benes [14] abordent aussi la creation de modèles 3D via la modélisation procédurale inverse, leur approche consiste à generer une représentation procédurale compacte, efficace et réutilisable d'un modèle architectural 3D polygonal : ils proposent le transfert de style, la synthèse de nouveaux bâtiments, la conversion du modèle en arbre et l'extraction de la grammaire.

Pour commencer, le modèle architectural polygonal est traité, décomposé et des règles terminales et non terminales en sont déduites : la sortie est une grammaire qui détermine les contraintes permettant de préserver le style.

La première étape consiste à organiser le modèle en arbre : la boite englobante est calculée et la racine est la boite englobante de toute la forme, le prochain plus grand composant est ensuite inséré dans l'arbre. L'insertion effectue une recherche descendante pour trouver le nœud parent, et repète l'étape autant de fois que necessaire. Cet arbre sera ensuite inspecté pour identifier les règles. Le modèle 3D peut être manuellement ou automatiquement segmenté et étiqueté à l'aide d'un algorithme de segmentation architecturale qui donne lieu à des composants,

La grammaire contient la définition et l'application des règles. De plus, tout motif régulier ou irrégulier peut être encodé comme un sous-arbre et

exporté comme une règle paramétrée. Pour étiqueter les règles, elles sont en premier lieu identifiées : ce sont les sous-arbres répétitifs et non répétitifs ensuite vient la découverte de motifs de répétition des règles de grammaire, après quoi la grammaire est exportée.

Les formes peuvent ensuite être éditées via une grammaire textuelle ou une interface graphique. L'édition permet de redimensionner en préservant la structure, peut diviser ou joindre un sous-arbre, insérer, remplacer, remplir ou dériver les règles existantes pour les modifier via la sélection d'un bâtiment vers la géométrie de destination.

2.2.4 Approche par les grammaires formelles

En 1972, Stiny et Mitchell sont les premiers à introduire les grammaires de formes. De la même manière que les L-systèmes, elles sont composées de règles de dérivation qui utilisent un vocabulaire qui est un ensemble fini de formes. A partir d'un forme d'axiome choisi dans le vocabulaire, un processus de dérivation permet de créer de nouvelles formes et modèles qui peuvent être extrêmement complexes et détaillés.

En 1978, Stiny et Mitchell [41] créent une grammaire paramétrique dont l'ensemble de règles permet de générer des plans pour les villas de Palladio. Suivant une étude de la géométrie de ces villas, ils ont conclu que le processus de génération se déroulait en 8 étapes : la définition des grilles, la création du mur extérieur, l'agencement des pièces, le ré-alignement des murs intérieurs, la création des entrées principales—portiques et inflexions des murs extérieurs, la création des ornements des colonnes extérieures, la création des fenêtres et portes, l'application des règles terminales. Ces règles ont pu être utilisées non seulement pour créer des villas fidèles à ce style particulier mais permettent également de déterminer qu'une villa appartient bien à ce style, dans ce cas : il doit exister une séquence de règles de cette grammaire aboutissant à sa création. Ils mettent alors en évidence le but derrière la caractérisations d'un style : clarifier la similitude sous-jacente de la structure, fournir les critères nécessaires pour déterminer si un bâtiment est une instance d'un style et fournir les outils pour concevoir de nouveaux bâtiments qui sont des exemples du style.

En 2001, Parish et Müller [35] ont commencé à modéliser les environnements urbains en utilisant des grammaires de formes où chaque bâtiment était composé de modèles de masse dont la géométrie et les détails de façade ont été générés par un L-système. Le système CityEngine se compose de plusieurs outils différents. Les données d'entrée environnementales tels que les cartes de densités de la région sont d'abord envoyées au système de génération de route, en utilisant un L-système étendu. Après l'obtention d'un réseau routier cohérent, les zones entre ces routes sont subdivisées pour définir les terrains à bâtir. Ensuite, en appliquant un autre L-Système associé à des opérateurs de transformation, les modèles de bâtiments sont générés en extrudant les empreintes au sol et en transformant le volume obtenu, ces derniers sont progressivement affinés. Un outil de création semi-automatique de façades utilise la stratification et une technique de composition pour finaliser le processus.

CityEngine est capable de modéliser un environnement urbain à grande échelle grâce à l'intégration des données géographiques et statistiques, ce qui permet de nécessiter une interaction limitée de l'utilisateur. Les modèles de masses sont générés en ajoutant et en raffinant une séquence de volumes verticalement et horizontalement.

Wonka et al. [46] ont ensuite démontré comment créer des détails plus convaincants sur les façades de chaque bâtiment en introduisant les "Split Grammars", un nouveau type de grammaire d'ensemble paramétrique avec un système de correspondance d'attributs orienté par une grammaire de contrôle. L'approche proposée par Wonka, Wimmer et Ribarsky [45] cible une dérivation complètement automatique des grammaires après que les objectifs architecturaux aient été définis. De nombreuses nouveautés sont présentées, notamment : la restriction des types de règles permises, ce qui rend cette grammaire puissante mais aussi simple d'utilisation, un système de correspondance de paramètres qui permet à l'utilisateur de spécifier plusieurs objectifs de conception pour guider la dérivation, enfin, ils introduisent les grammaires de contrôle : des grammaires simples sans contexte qui traitent de la distribution spatiale des styles de conception d'une manière qui correspond aux principes architecturaux. A partir d'une forme initiale, la grammaire génère les façades du bâtiment, qui sont divisées en éléments structurels, jusqu'aux détails comme les seuils de fenêtre, corniches etc (voir figure 2.7). Pour modéliser une grande variété de bâtiments, il est nécessaire qu'il y ait un

choix parmi plusieurs règles d'appariement durant la dérivation à la fois dans le "Split Grammar" et dans la grammaire de contrôle. L'approche choisie est de créer une base de données de règles de grammaire qui permet de générer de nombreuses possibles conceptions avec cette base de données uniquement. La sélection d'une règle doit être fait de telle manière que la dérivation produise des résultats cohérents et plausibles, fournisse une variation suffisante et que les critères de conception spécifiés par l'utilisateur soient respectés. Le système obtenu est très flexible car une fois qu'une grammaire de base a été créée, l'utilisateur est libre d'introduire des données supplémentaires sous la forme d'attributs ou de règles afin de guider la dérivation.

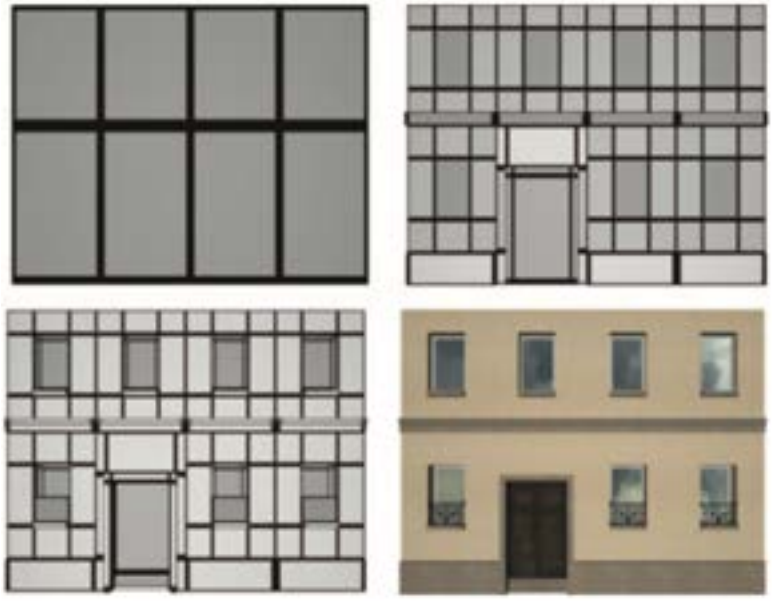

Figure 2.7 – Exemple de dérivation d'une façade. Source : Wonka et al. [46]

Müller et al. [32] ont complété ces concepts avec la grammaire de forme CGA (Computer Generated Architecture) [32] : un langage de programmation spécifique pour générer du contenu 3D architectural qui s'appuye sur les *split grammars* et sera intégrée à CityEngine. CGA est une solution extrêmement complète pour générer des extérieurs de bâtiments détaillés provenant de modèles de masse complexes. Un modèle détaillé de la ville de Pompéi a pu être complètement recréée avec ce système (voir figure 2.8). L'idée consiste

à définir des règles qui affinent itérativement un design en créant de plus en plus de détails. Ces règles opèrent sur des formes qui se composent d'une géométrie dans une boîte délimitée localement. Une grammaire de contrôle s'ajoute à ce moteur pour générer les variations sur le modèle. Les principales contributions se trouvent au niveau du contrôle offert à l'utilisateur au cours des dérivations avec l'ajout d'attributs prioritaires, l'introduction de l'opérateur "component split" qui fait le lien vers la décomposition du volume 3D en façades, et la sensibilité au contexte qui permet à la grammaire de s'adapter à la taille des primitives qui émergent.

Figure 2.8 – Différentes vues du modèle de Pompéi. Basée sur de véritables empreintes de bâtiments, la ville a été générée avec 190 règles de forme CGA écrites manuellement. Source : Müller et al. [32]

Pour poursuivre sur les grammaires de formes et leur intégration dans le processus de conception, Halatsch, Kunze et Schmitt [20] analysent également la grammaire de formes CGA de Müller et al. [31] et ont étendu ce système vers un framework de visualisation pour les architectes en offrant la possibilité de décrire des modèles de conception urbaine sous la forme de modèles hiérarchiques. Dans leur analyse, ils mettent en avant la possibilité d'implémenter une méthode de visualisation dans un pipeline de génération existant. Les formes CGA et les styles désirés peuvent notamment être

stockées dans des bases de données pour une réutilisation des modèles. Ils proposent également de nombreux domaines d'applications, notamment la reconstruction des édifices, la pré-visualisation des grandes agglomérations urbaines et l'évaluation des critères de simulation à l'échelle urbaine, grâce à la capacité de dériver automatiquement des modèles 3D de haute qualité pour une visualisation très détaillée et rapide de modèles de villes.

Koutsourakis et al. [26] utilisent une autre approche, également basée sur les grammaires, qui consiste à modéliser des styles architecturaux en s'appuyant sur des formes basiques et un ensemble de règles paramétriques. Cette méthode se focalise sur les façades, et analyse la géométrie et sémantique des façades en combinant les grammaires de formes avec les champs aléatoires de Markov (MRF). L'ensemble du pipeline utilise comme entrée une image rectifiée du bâtiment et une grammaire de forme paramétrique. Partant du fait qu'une façade peut être définie en utilisant les grammaires de formes en utilisant des paramètres spécifiques, l'instanciation des paramètres crée une grammaire spécifiques qui peut générer un bâtiment 3D unique. L'objectif est de calculer les paramètres inconnus permettant de faire correspondre l'édifice généré et l'image fournie. Les règles sont formulés en tant que champ aléatoire de Markov. L'énergie minimisée par le MRF repose principalement sur des mesures de similarité entre les sous-régions de la façade et d'autres caractéristiques. L'inférence est déterminée par l'algorithme Belief Propagation garantissant la solution optimale. Cette approche automatique lie la segmentation 2D et la reconstruction 3D. Elle examine une image de façade rectifiée et l'adapte un arbre hiérarchique. Cette tâche est formulée comme un champ aléatoire de Markov, où l'arbre de l'image de façade est convertie en une grammaire de formes chargée de générer un modèle procédural.

Enfin Vanegas, Aliaga et Benes [43] abordent la reconstruction à partir des bases de données cartographiques et de navigation existantes pour créer des bâtiments en 3D. Les bâtiments de type Manhattan World (MW) peuvent être encodés en grammaires décrivant les transitions entre les étages. La méthode utilise une grammaire de bâtiment type MW ainsi que l'enveloppe du bâtiment 3D pour la reconstruire progressivement. Elle exploite la cohérence entre les étages d'un bâtiment, puis effectue la transition vers les étages de manière à correspondre à la vue aérienne, ce qui donnera lieu à un modèle 3D amélioré avec des règles de réécriture pour les transitions d'un étage à l'autre.

Les bâtiments MW sont représentés par une séquence d'étages, chacun formé par une séquence de faces qui sont des quadrilatères. La représentation d'une nouvelle forme d'étage est basée sur la forme de l'étage précédent en appliquant le changement. Cette modification est capturée par une règle de réécriture. Le changement de forme d'un étage à l'autre peut être obtenu par une ou plusieurs transitions appartenant à l'un des trois types suivants : en forme de L, en forme de U ou en refoulement.

La grammaire impose une structure cohérente et plausible par des contraintes inter-étages telles que : les formes d'étages sont fermées, la ligne définissant un étage ne doit pas s'auto-intersecter, les transitions qui modifient significativement la forme d'un étage sont limitées. Le processus commence par la boite englobante de l'empreinte, et la hauteur. La plupart des transitions entre étages pourraient être encodées dans une seule règle paramétrée de réécriture.

Avec cette représentation du bâtiment, la méthode de reconstruction de bâtiment consiste à modifier chaque chaîne d'étage de manière à améliorer une mesure de cohérence entre le modèle de bâtiment et les images capturées. en utilisant des signaux qui contiennent des valeurs géométriques aux tournants de l'empreinte au sol, et en faisant correspondre les signaux : une faible corrélation qui déclenchera l'application des règles de réécriture.

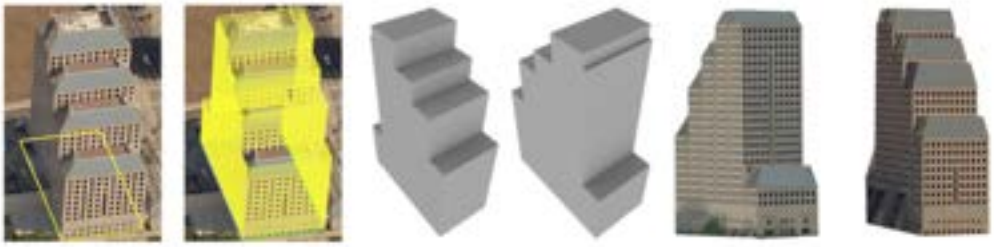

Figure 2.9 – Reconstrucion d'un bâtiment. Source : Vanegas, Aliaga et Benes [43]

2.2.5 Approche par les graphes

Dans Martin [29], les auteurs considèrent une toute autre approche pour la création de plans au sol, en abordant le problème à partir de l'intérieur d'une maison. En effet, alors que la plupart des autres méthodes génèrent des extérieurs d'édifices sans structure ni aménagements intérieurs, leur méthode commence par l'intérieur et utilise la structure intérieure pour dicter l'aspect extérieur.

La phase de génération du graphe commence par la détermination de la structure des salles publiques. La porte avant de la maison est ajoutée en premier et la production utilise une grammaire sans contexte. Cette étape est responsable de la structure, non pas de la sémantique et les pièces n'ont pas encore de type spécifique. Le type de pièce est attribué durant l'étape suivante (salon, salle à manger, etc) à chaque salle publique en utilisant des statistiques qui peuvent être définis par l'utilisateur. Ceux-ci garantissent la création d'espaces plus réalistes avec l'addition de la grammaire. L'étape suivante place au hasard les salles privées qui rejoignent les salles publiques jusqu'à remplir tout l'espace qui leur est dédié. La dernière étape consiste à ajouter les dernières pièces qui peuvent être ajoutées rapidement et sans impact sur les autres pièces déjà existantes.

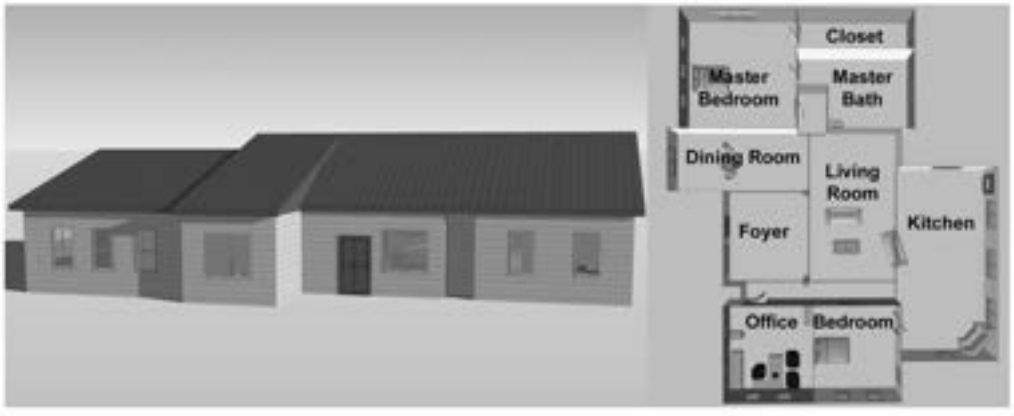

Figure 2.10 – À gauche : maison basée sur le plan d'étage généré par l'algorithme. À droite : vue de haut du même plan. Source : Martin [29]

Le graphe créé est un arbre dont la racine est la pièce adjacente à la porte d'entrée. La taille correcte des pièces est obtenue en utilisant une méthode

de Monte-Carlo pour choisir quelle pièce croître ou de réduire. Chaque pièce du graphe exerce une pression extérieure proportionnelle à la taille de la pièce par rapport aux autres pièces. L'architecture et la théorie des graphes combinées ont permis de créer une grammaire qui construit un graphe où chaque noeud correspond à une pièce et chaque arête à une connexion entre des pièces. Les intérieurs résidentiels ont été obtenus avec un haut degré d'automatisation grâce à la combinaison des grammaires et statistiques. La figure 2.10 montre un exemple de maison générée par l'algorithme.

2.2.6 Approche par un système multi-agents

Guo et Li [19] proposent une solution à la disposition automatique des formes, aux dimensions et aux positions des espaces internes pour satisfaire aux critères architecturaux à l'aide des agents. Ces travaux présentent une méthode pour la génération automatique d'une disposition architecturale en combinant un système de recherche de topologie multi-agents et un processus d'optimisation évolutif.

Un système multi-agent génère un agencement de la topologie pour une optimisation ultérieure. Les pièces sont représentées avec des agents de différents types (bulle, capsule) dont la taille et/ou l'orientation peuvent être affectés via l'interaction avec d'autres agents. Des règles d'attraction, de répulsion, d'échange et de compression sont appliqués pour modifier les positions et l'orientation des agents.

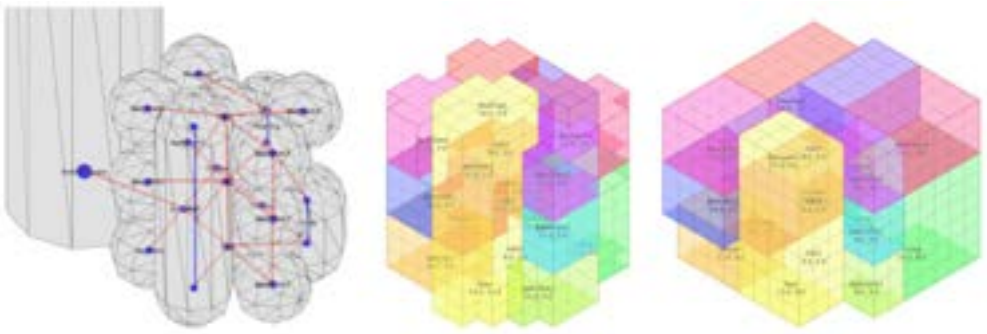

Figure 2.11 – A gauche : l'agencement spatial généré, à droite : conversion de cet agencement en système de grilles. Source : Guo et Li [19]

La seconde étape est l'affinement de l'agencement obtenu pour satisfaire des critères architecturaux prédéfinis. Les agents sont convertis en un système de grille en appliquant le principe du diagramme de Voronoï. La figure 2.11 permet de voir ce passage de l'agencement spatial des formes vers un système en grille.

2.2.7 Approche par l'utilisation de pinceaux et la création d'esquisses

Benes et al. [2] ont créé un outil qui permet de modifier une disposition urbaine existante : l'édition est contrôlée par l'utilisateur à l'aide d'un ensemble de pinceaux qui appliquent des opérations localisées tels que la relocalisation des emplois et des habitants. L'algorithme prend en entrée la hauteur du terrain, le réseau routier, les parcelles, les emplois et la population des bâtiments, chacun ayant sa propre représentation telle que : grille, graphique, polygone ou maillage 3D. Les terrains en particulier sont caractérisés par un contour, les bâtiment qu'ils contiennent, le nombre d'emplois et la population qu'il peut accueillir.

Lors de l'édition de la ville, la modification va préserver le style de la ville, maintenir les emplois et la population à des niveaux constants, et maintenir la cohérence du réseau routier en rendant chaque quartier accessible.

Des opérations atomiques peuvent être appliquées tels que le transfert qui permet de relocaliser les emplois et les personnes entre deux bâtiments, tout en maintenant la validité du lot. La géométrie des lots peut être fusionnée ou scindée tout en conservant une cohérence en ce qui concerne les emplois et la population, les blocs de terrain peuvent également être fusionnés et divisés. Ces opérations atomiques sont combinées dans des brosses paramétriques qui modifient localement la ville pour repousser, attirer ou relocaliser le volume ou la population de la ville, elles déplacent ces derniers vers la zone d'infuence qui est un cercle où le pinceau peut apporter des modifications sans modifier la géométrie de la route et des blocs. Des paramètres contrôlent leur influence sont tels que : la région d'impact, la région d'influence, la population, la hauteur, si la fusion ou la division est autorisée, la distance de temps de parcours.

La brosse de préservation de style s'assure que le pinceau n'altère pas la

cohérence visuelle en capturant le style d'aménagement urbain, cela se fait en catégorisant les intersections et les statistiques de distribution et d'orientation des types d'intersection, des angles de jonction et des distances entre eux. Ensuite, les blocs vides sont reconstruits en générant les artères, puis les rues. Au fur et à mesure que chaque route est développée, elle tente de se connecter et reproduit le style précédent en échantillonnant les distributions.

Les modifications à grande échelle pouvant conduire à des incohérences, chaque incohérence est répertoriée par ordre en commençant par celle qui enfreint le plus la validité de l'aménagement, puis la liste est traitée dans cet ordre : éliminer toutes les routes à forte pente, utiliser l'opération de fusion de blocs pour corriger les blocs, le pinceau reconstruction est automatiquement appelé pour remplir les parties vides.

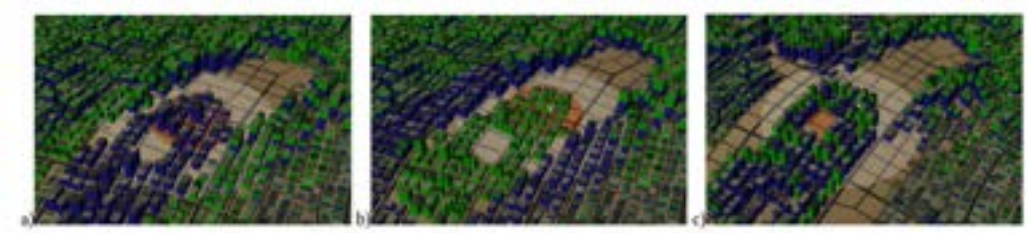

Figure 2.12 – Utilisation du pinceau d'attraction pour attirer : (a) la population uniquement (bleu), (b) les emplois uniquement (vert), (c) ou les deux Source : Benes et al. [2]

Enfin l'apprentissage automatique est utlisée par Nishida et al. [33] pour fusionner la modélisation basée sur l'esquisse et la modélisation procédurale. En se servant de grammaires, les règles sont automatiquement déduites de l'esquisse. Comme dans un dessin classique, ce système va décomposer le processus de création de modèles hiérarchiques en partant d'un stade grossier et en affinant au fur et à mesure.

Les types d'objets tels que le toit, les fenêtres, les formes de bâtiment sont définis dans plusieurs extraits de grammaire, l'utilisateur dessinera jusqu'à ce qu'une correspondance soit trouvée avec un extrait. La construction d'une variété de bâtiments est faite en combinant ces extraits. Une collection de réseau de neurones convolutifs est formée en tant que prétraitement avec les

images rendues pour chaque variété de bâtiment obtenue en échantillonnant au hasard les valeurs des paramètres de l'extrait. Des variations sur le modèle final peuvent être obtenues de manière procédurale.

Au moment du dessin, le réseau de neurones convolutifs identifie l'extrait et ses paramètres pour expliquer l'esquisse actuelle, puis ce réseau reçoit l'esquisse de l'utilisateur pour récupérer l'extrait de code et les valeurs de paramètre souhaités, enfin ils sont mis en correspondance et complétés avec du contenu généré de manière procédurale. La grammaire de sortie est composée d'extraits de grammaire qui, une fois combinés ensemble représentent le modèle 3D.

Dans certains cas, l'utilisateur peut appliquer un type d'objet de manière répétitive (par exemple, une rangée de fenêtres, une pile d'étages). Au lieu d'esquisser l'objet plusieurs fois ou en faisant un usage intensif du copier-coller, cet outil fournit un fournir un mécanisme rapide pour subdiviser la région, esquisser un type d'objet et générer des répétitions.

2.3 Représentation des villes

Dans la littérature, de nombreux modèles conceptuels ont déjà été proposés, car les besoins pour la gestion des villes, la navigation et la nécessité d'optimiser le stockage des données ont donné lieu à la création de standards. Plusieurs travaux vont plus loin dans la simulation d'une ville et prennent en compte les dimensions économiques et topographiques mais aussi la croissance de la ville dans le temps, permettant aux utilisateurs d'interagir avec le simulateur et de donner un grand nombre d'informations pour voir comment la ville va évoluer.

Des systèmes de génération de villes très réalistes ont pu être développés par les laboratoires de recherches au cours de ces dernières années. Leur complexité augmente plus le nombre de règles et de paramètres pris en compte augmente. Les modèles récents sont plus complexes et comportent des aspects géométriques et topologiques, conservant ainsi l'intégrité spatiale avec des classes thématiques. De nouveaux concepts ont été développés dans ce sens, ils se sont établis dans la norme ISO 19107 qui définit les opérations spa-

tiales standard à utiliser pour l'accès, la requête, la gestion, le traitement et l'échange de données d'informations géographiques pour des objets spatiaux (géométriques et topologiques).

2.3.1 CityGML

L'un des standards les plus connus s'appelle CityGML. CityGML est un modèle d'information communément utilisé pour les environnements urbains 3D. Il se base sur le format XML pour le stockage et l'échange de modèles d'environnements virtuels pour représenter des villes existantes avec des données SIG. Il comprend un module principal, qui définit les concepts basiques de l'environnement urbains et plusieurs modules thématiques. Ce système a été testé pour différentes applications comme l'environnement, l'ingénierie ou simplement pour les GPS. CityGML est également adapté pour le web 3D, ce qui devient de plus en plus important en ce moment.

Dans CityGML, une ville comprend les structures construites, l'élévation, la végétation, les plans d'eau, le mobilier urbain, etc. Les hiérarchies entre les classes thématiques, les agrégations, les relations entre les objets et les propriétés spatiales sont incluses dans le modèle. Ces informations thématiques permettent d'utiliser des modèles de villes virtuels pour des tâches d'analyse dans différents domaines d'application tels que les simulations, l'exploration de données urbaines et les enquêtes thématiques. Pour des domaines d'application spécifiques, CityGML fournit également un mécanisme d'extension pour enrichir les données avec des caractéristiques identifiables tout en préservant l'interopérabilité sémantique.

2.3.1.1 Concepts généraux

CityGML couvre les aspects géométriques, topologiques et sémantiques des modèles urbains, tel que le montre la figure 2.13. Les classes peuvent représenter les bâtiments, végétaux, plans d'eau et les installations de transport comme les réseaux routiers. Les propriétés spatiales et sémantiques sont structurées en cinq niveaux de détail (LoD) consécutifs, où LoD0 définit un modèle régional grossier et le LoD4 le plus détaillé comprend les intérieurs de bâtiments. Le modèle de bâtiment permet de représenter les aspects thématiques et spatiaux des bâtiments, des éléments de construction et des

accessoires en quatre niveaux de détails.

La description des entités thématiques englobe des attributs, des relations et des hiérarchies imbriquées entre les entités. Les objets géométriques sont affectés à des entités, de ce fait, une hiérarchie d'agrégation se fait en même temps au niveau sémantique et géométrique. Ces informations sémantiques peuvent être utilisées pour des requêtes thématiques, des analyses ou des simulations.

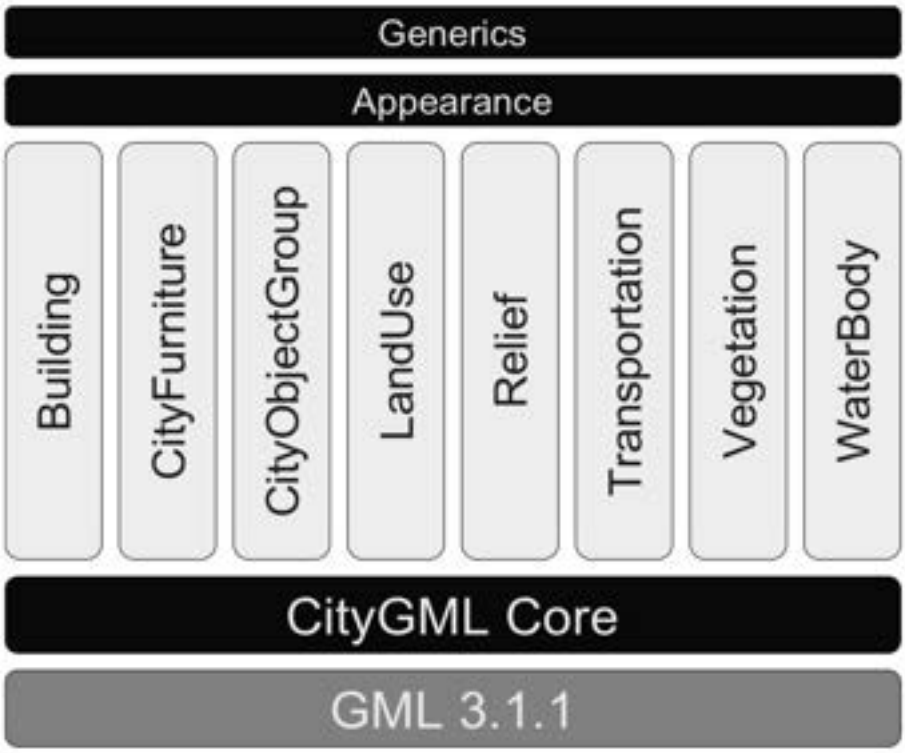

Figure 2.13 – Modularisation de CityGML 1.0.0. Source : Kolbe [24]

En plus des informations sémantiques et de la géométrie, les entités peuvent avoir des apparences, c'est-à-dire des informations sur les propriétés observables d'une entité. Les apparences peuvent représenter des textures et des matériaux, mais ne se limitent pas aux propriétés visuelles. Les apparences peuvent transporter n'importe quel quantité comme le rayonnement

infrarouge, la pollution sonore, etc.

2.3.1.2 Niveaux de détails

CityGML supporte cinq niveaux de détails, illustrés dans la figure 2.14, dans le cas de différents processus de collection de données pour des applications différentes.

LOD0 : est une représentation 2.5D de l'empreinte des bâtiments.

LOD1 : les bâtiments sont représentés comme des bâtiments en blocs, c'est-à-dire des formes 3D simples qui peuvent être obtenues en extrudant des polygones d'empreinte horizontale le long de l'axe vertical. Les façades et les toits peuvent être texturés dans tous les niveaux de détail.

LOD2 : représente un modèle avec un toit qui est modélisé comme un toit normalisé. Les bâtiments peuvent avoir une géométrie de toit explicitement spécifiée, décrite par un ensemble de polygones en 3D. De plus, les murs extérieurs et le toit d'un bâtiment sont représentés comme des objets thématiques distincts.

LOD3 : est un modèle architectural détaillé avec des ouvertures telles que des fenêtres et des portes, qui sont accessibles en tant qu'objets thématiques séparés. Toutes les parties d'un bâtiment peuvent être texturées.

LOD4 : complète un LOD3 en incluant des éléments intérieurs (Kolbe, 2009) tels que les murs intérieurs, les pièces et les meubles sont fournis en tant qu'objets thématiques séparés.

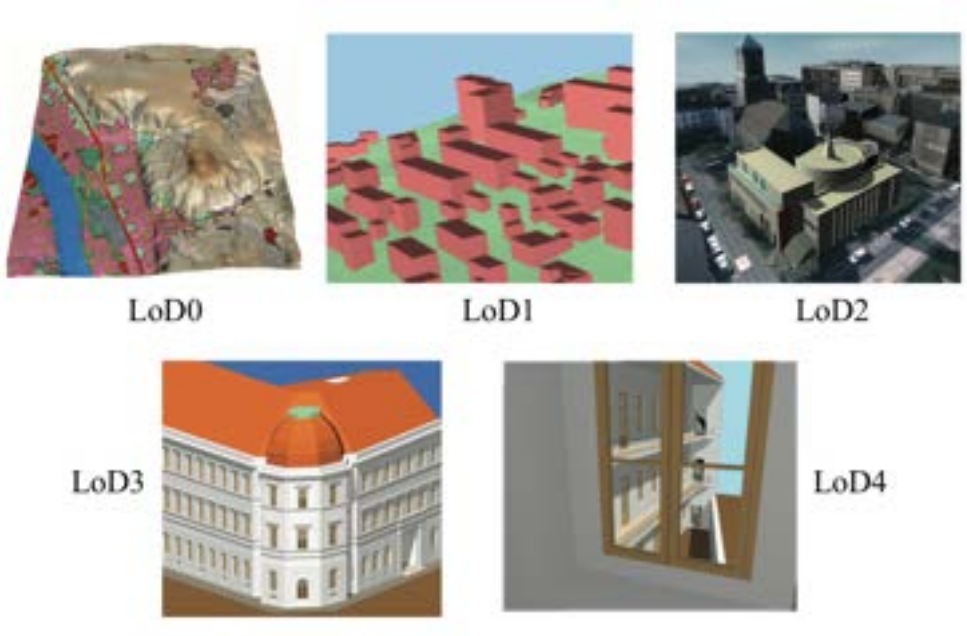

Figure 2.14 – Niveaux de détails. Source : Kolbe, Gröger et Plümer [25]

Un même bâtiment peut contenir des représentations de plusieurs niveaux de détail simultanément. La géométrie des bâtiments et des parties des bâtiments est spécifiée à l'aide d'un modèle géométrique de CityGML, qui est également utilisé pour d'autres composants de modèle de ville. Dans le modèle géométrique, un objet 3D est décrit par des agrégations hiérarchiques de polygones qui représentent les surfaces limites de l'objet. Le modèle prend en charge la spécification d'images de texture et des coordonnées de texture pour toutes les surfaces.

2.3.1.3 Modélisation sémantique-géométrique cohérente

CITYGML implémente une modélisation cohérente entre les aspects sémantiques et les propriétés géométriques et topologiques. Les modèles sont représentés avec ces deux hiérarchies différentes où les géométries sont affectées aux objets sémantiques représentant les entités du monde réel. Chaque modèle comprend des relations et des hiérarchies d'agrégation entre les objets.

2.3.2 3DCityDB

3DcityDB est un logiciel open source, qui permet d'importer des données CityGML dans une base de données 3D telle que ORACLE Spatial ou PostgreSQL/PostGIS. Ce logiciel est capable de supporter tout type de niveau de détails et d'apparences, des modèles complexes de terrain et de construction. Il permet également d'importer, gérer, analyser, visualiser et exporter des modèles de villes 3D virtuelles suivant le standard CityGML. Son architecture est montrés à la figure 2.15.

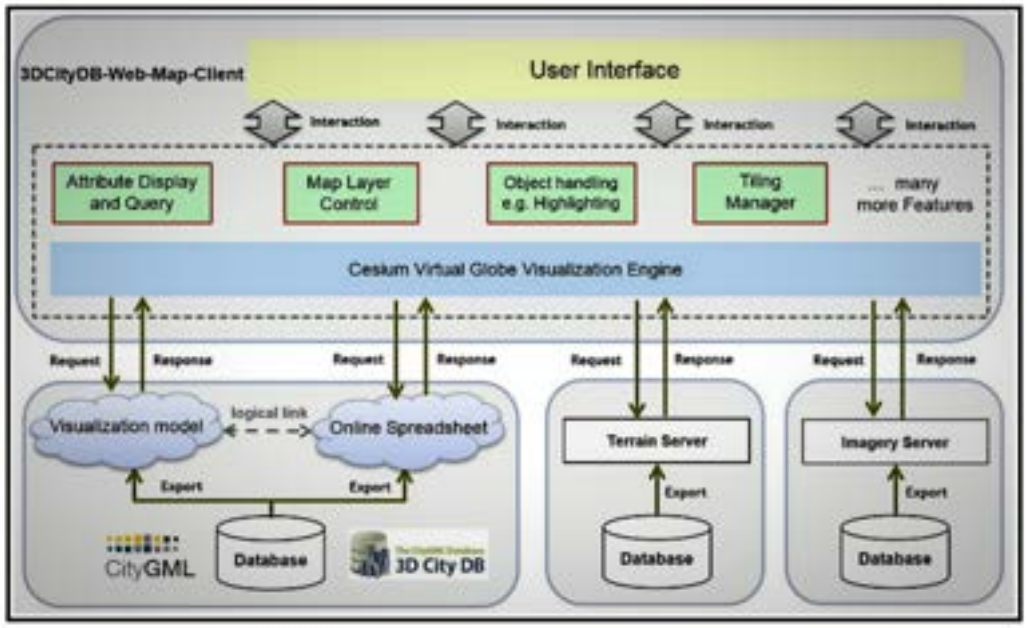

Figure 2.15 – Architecture de 3DCityDB. Source : Yao et al. [47]

3DCityDB est implémenté comme schéma de base de données relationnelle avec des procédures stockées pour différents systèmes de gestion de bases de données relationnelles. Il est livré avec un ensemble d'outils logiciels basés sur Java pour fournir un accès aux données Web, l'import et l'export d'ensembles de données CityGML, et pour générer des modèles de 3D avec des données thématiques. Un client Web 3D basé sur HTML5/WebGL pour l'exploration interactive des modèles de visualisation 3D générés est inclus. Grâce à l'utilisation combinée de ces composants logiciels, des chaînes de

travail complètes peuvent être établies. Celles-ci vont de la lecture, du traitement et de l'écriture du contenu du modèle de ville 3D, à la visualisation et à l'exploration de données interactives sur le Web, dans une visionneuse de carte 3D.

2.3.3 Keyhole Mark-up Language

Keyhole Mark-up Language (KML) est un format de fichier basé sur XML pour représenter les données géographiques sur des applications Web telles que Google Earth et Google Maps. KML est partiellement similaire à Geography Markup Language (GML). On peut y définir un ensemble d'entités comme des modèles 3D, des polygones, des points d'intérêts ou des images. KML permet d'afficher, placer des traçages, augmenter des points spécifiques avec des informations supplémentaires, ou encore dessiner des objets arbitraires sur une carte qui prend en charge le langage.

2.3.4 Industry Foundation Classes

Le format IFC est un format neutre de données pour décrire, échanger et partager des informations généralement utilisées dans l'industrie du bâtiment et de la construction. IFC est la norme internationale pour openBIM, il a été imaginé pour fournir une base universelle pour l'échange de données concernant les bâtiments.

Le schéma IFC codifie : la sémantique, les attributs (comme la matière ou la couleur), les relations entre objets et processus (installation, opérations)... La spécification du schéma peut décrire comment une installation est utilisée, comment elle est construite et comment elle est exploitée. IFC peut définir les composants physiques des bâtiments, ainsi que des modèles d'analyse structurelle plus abstraits.

Le modèle de données IFC est un modèle de données orienté objet basé sur des définitions de classe représentant les objets (éléments, processus, formes) qui sont utilisés par des applications logicielles au cours d'un projet de gestion de construction. On y retrouve par exemple les entités comme IfcWall pour les murs ou des représentations de géométries dans IfcExtrudedArea-

Solid pour l'extrusion de solides.

Le modèle IFC est représenté par des structures qui renferment des espaces (IfcSpatialStructureElement). Les structures qui en dérivent sont les sites, les buildings, les étages. La plupart des géométries utilisées pour les éléments de construction sont des modèles solides définis en tant que IfcProduct, dans lequel tous les points intérieurs sont connectés. Les modèles solides utilisés peuvent être :

1. La représentation de frontières avec les facettes : dans laquelle toutes les faces sont planes et toutes les arêtes sont des lignes droites.
2. Une zone de solide extrudée : définie en balayant une surface plane délimitée. Cette zone plane peut contenir des limites internes.
3. La Géométrie solide constructive : une collection de solides primitifs, combinés à l'aide d'opérations booléennes.

Le champ d'application du modèle IFC est limité aux bâtiments et aux sites, il n'y a donc pas de classes d'entités topographiques telles que le terrain, la végétation, ou les plans d'eau. Ceux-ci doivent être modélisés dans IFC en tant qu'objets génériques.

2.3.5 QUASY

QUASY [3] est un modèle sémantique créé pour la modélisation des bâtiments. Il est structurellement similaire à CITYGML et supporte plusieurs niveaux de détails, sa complexité se situe entre ce dernier et le modèle IFC. Pour ce modèle, un bâtiment est un QuBuilding, et il introduit le Quvariant : un concept flexible et généralisé qui peut contenir les informations pertinentes via 3 types d'attributs : géométriques, sémantiques et paramétriques (qui peut être utilisé pour instancier les modèles géométriques et des composantes de bâtiments). Contrairement à CITYGML, QUASY représente la topologie des pièces et assigne les composantes internes à chaque pièce et se focalise sur l'utilisation des objects sémantiques. Benner, Geiger et Leinemann [3] abordent également la transformation de données CAD vers le modèle QUASY, plus précisément le modèle IFC, quelques données thématiques ont été ignorés car non pertinent pour un modèle de ville. Les contenus d'un modèle IFC peuvent y être affichés et les données géométriques manipulées après extraction des données sémantiques, des géométries et execution

des opérations sur les formes, le tout jusqu'à ce que les éléments d'un bâtiments soient des facettes correspondant à l'enveloppe externe des bâtiments.

2.3.6 Base de données pour la ville de Berlin

Des travaux dans le domaine de la planification urbaine [39] se sont penchées sur la création d'une base de données urbaine 3D basée sur le format d'échange urbain CityGML. Le modèle proposé est riche sémantiquement, géométriquement et structuré hiérarchiquement. Pour se faire, le modèle CITYGML a été plus ou moins reproduit dans une base de données relationnelle Oracle. Cette nouvelle base de données implémente les caractéristiques clés de CITYGML tels que les différents niveaux de détails, les géométries flexibles, la modélisation thématique complexe. Cependant, pour stocker de manière efficace un grand volume de données, le modèle est simplifié sur deux principaux aspects : le traitement des récursions où chaque objet stocke alors son élément parent et son élément racine, et sur la simplification des classes géométriques avec l'introduction d'une classe abstraite (BRepGeometry) qui sera la base de tout objet géométrique contenant des surfaces. Pour le processus d'import-export de données CITYGML, chaque tâche est traitée parallèlement par des treads séparés pour achever un traitement rapide des données.

2.4 Évaluation

2.4.1 Dans la représentation de bâtiments

Nous avons pu voir de nombreuses représentations de villes, dans le domaine spécifique des systèmes d'information géographique, diverses bases de données ont été développées pour stocker des informations spatiales. Ces outils ont tendance à se concentrer sur de grandes données géographiques 2D plutôt que sur le traitement des données individuelles. L'une des plus importantes étant CityGML qui est un format extrêmement complet mais également lourd et complexe, cependant, les concepts qu'il introduit restent pertinents.

CityGML est un format relativement nouveau et malgré de nombreux avantages, sa prise en charge et son adaptation logicielle sont encore infé-

rieures à celles des formats d'infographies tels que COLLADA. Cette limitation, couplée au manque général de données multi-LOD, rend les données multi-LOD CityGML pratiquement inexistantes dans la pratique (Biljecki et al., 2015b).

2.4.2 Dans la création de bâtiments

Un certain nombre de facteurs entrent en jeu dans la création de modèles urbains 3D, tels que l'application, le rendu attendu, le budget, la période et la superficie à couvrir. On retrouve néanmoins toujours ces éléments dans la construction des modèles : le degré de réalité, les types de données en entrée, et le degré de fonctionnalité. La variété démontrée pour chaque facteur montre la diversité dans le domaine actuel de la modélisation.

Il y a principalement deux approches émergeant de la diversité de méthodes de modélisation :

— Utilisation de différents progiciels et d'outils de CAO, d'outils de création de rendu d'image, de base de données et d'interface. Cette approche convient au développement d'un modèle propriétaire pour une petite zone et des modèles architecturaux en mettant l'accent sur l'effet visuel plutôt que sur sa fonctionnalité.
— Intégration de SIG avec visualisation 3D : extrusion d'éléments verticaux sur des données cartographiques numériques occasionnellement combiné avec des techniques de correspondance d'images avec des données de photographie aérienne. Cette approche est plus appropriée pour couvrir les grandes surfaces ainsi que l'analyse spatiale et les simulations.

À l'heure actuelle, le moteur de modélisation procédural le plus avancé est le CityEngine d'ESRI, largement utilisé par les urbanistes et autres professionnels. Bien que les grammaires de formes CGA utilisées par ESRI puissent générer des modèles de construction visuellement convaincants, Finkenzeller et Bender notent qu'ils manquent d'informations sémantiques concernant le rôle de chaque forme dans le bâtiment complet. Jusqu'à présent, les efforts de modélisation procédurale ne semblent pas s'être beaucoup concentrés sur les le lien entre SIG et CAO. Considérant qu'un moteur procédural peut être programmé pour produire des données avec une granularité spécifique,

nous abordons les deux idées dans notre travail en définissant différentes procédures pour générer des bâtiments dans une série de représentations différentes.

Dans le domaine des grammaires en particulier, beaucoup de celles que nous avons explorées se concentrent sur la reproduction d'un style spécifique, dans le cas où la structure ciblée est plus générale, les règles sont compliquées et difficiles à reproduire. Bien sûr nous ne pouvons pas recréer un système comme CityEngine d'ESRI, nous nous sommes penchés sur la capacité à reproduire de nombreux styles tout en offrant une alternative plus intuitive.

Nous gardons les idées suivantes comme les plus pertinentes à notre étude dans la liste de Chau et al. [9] qui tente de caractériser une grammaire de forme idéale dans le cadre de la prise en charge de la conception géométrique des produits de consommation :

— Offrir des formes tridimensionnelles.
— Intégrer une interface utilisateur intuitive.
— Fournir des mesures esthétiques pour classer les designs.
— Offrir des surfaces de support et des solides pour le modèle.
— Fournir une interprétation sans ambiguïté des designs résultants jusqu'à leur réalisation physique.

2.4.3 Dans la création de réseaux routiers

L'un des premiers systèmes à générer des villes virtuelles entières a été présenté par Parish et Mueller [2001]. L'utilisateur fournissait plusieurs types de couches d'entrée pour décrivant un environnement sous-jacent et un ensemble de grammaires écrites personnalisées basées sur des L-systèmes sensibles à l'environnement sont utilisés pour générer des tracés de rues et des bâtiments simples. Leur travail n'inclut pas la modélisation comportementale, mais indique comme futur souhaitable de travailler à l'inclusion de schémas comportementaux dans le mécanisme de création de routes. Une seule direction de communication est fournie entre les couches d'entrée et les résultats générés et il n'y a pas d'interdépendance entre les couches d'entrée. Bien que les couches puissent être modifiées manuellement pour des interdépendances simples, elles doivent être connues du concepteur et ne s'adapte pas bien aux grands modèles.

Des recherches ultérieures se sont concentrées sur l'amélioration d'aspects particuliers de la génération urbaine. Par exemple, le processus de génération d'un réseau routier grâce à la manipulation interactive de champs tensoriels a été abordé par Chen et al. [2008]. Bien que ces méthodes puissent produire une variété de configurations de routes, les processus de conception ne tiennent pas compte à la fois de la répartition de la population et des emplois et de l'accessibilité entre les différentes parties du modèle. Une telle considération nécessite une édition manuelle (pour [Chen et al. 2008]) ou de nombreux exemples d'entrées bien choisis (pour [Aliaga et al. 2008]).

2.5 Discussion et analyse

Suivant notre étude de l'état de l'art, les problématiques suivantes en émergent :

1. La grande majorité des méthodes de modélisation procédurale sont spécialisées dans la génération d'un type spécifique de contenu ou de fonctionnalité. Pour être utile dans la conception de mondes virtuels complets, tout le contenu hétérogène généré devra être ajusté, assemblé et maintenu ensemble dans le moteur.
2. Les méthodes procédurales sont souvent définies par un ensemble de règles et de paramètres non intuitifs, qui peuvent être difficiles à comprendre et nécessitent une connaissance approfondie des éléments internes du système et de la technique utilisée. Leur effet sur le résultat final n'est pas toujours clair et prévisible. Une petite modification de la valeur d'une règle ou d'un paramètre peut provoquer une réaction en chaîne des modifications qui se propage dans tout le modèle. En conséquence, les méthodes procédurales permettent généralement un contrôle utilisateur plutôt limité.
3. D'autre part la contrepartie à payer pour l'expressivité des grammaires est la difficulté et le temps nécessaire à la création d'un ensemble de règles, cela justifie la nécessité d'automatiser la procédure de modélisation basée sur les grammaires mais il faudrait trouver un juste milieu entre autonomie totale qui est fastidieuse pour l'utilisateur et l'automatisation totale du processus qui lui ôtera le contrôle sur le résultat attendu.

4. Dans le domaine du BIM, la création de modèles tels que l'IFC de manière procédurale et l'intégration d'informations contextuelles via les grammaires n'est pas particulièrement exploitée.
5. Une base de connaissance d'ensemble de règles par lieu est nécessaire pour faire le lien avec les SIG et d'offrir un contrôle à différents niveaux de la modélisation
6. Le défi est de faire coopérer activement différentes méthodes de création dans la génération d'un modèle complexe. L'utilisation de techniques adaptées tout en préservant les qualités individuelles de chaque méthode permettrait de générer chaque élément du modèle de manière cohérente et constructive.

Les techniques précédemment développées fournissent les concepts utiles pour appréhender les problématiques de la thèse et le développement de structures adaptées pour travailler sur les données architecturales et géographiques. De nombreux efforts de recherche ont été consacrés à l'augmentation de la facilité d'utilisation de la modélisation procédurale, en particulier pour améliorer la qualité et le degré de contrôle des utilisateurs, en raison de son rôle crucial de facilitateur pour les professionnels créatifs non techniques. Des résultats significatifs ont été obtenus jusqu'à présent dans ce sens. L'interopérabilité des méthodes proposées sera garantie par l'existence de standards adaptés comme IFC.

Chapitre 3

Modélisation de bâtiments

3.1 Le système

3.1.1 Vue d'ensemble du système

Dans ce qui suit nous allons expliquer le fonctionnement de notre système. Ce système permet, en partant d'une ou plusieurs empreintes au sol, et éventuellement avec des informations géographiques, de générer semi-automatiquement un ou plusieurs bêtiments au format IFC avec des apparences spécifiques (correspandant à leur emplacement géographique) et divers niveaux de détails. Nous commencerons par une introduction sur les grammaires, puis nous aborderons la création et le fonctionnement de notre grammaire et de l'interpréteur de grammaire. Les alternatives pour la création automatique de règles seront présentés ensuite et enfin l'export vers un fichier IFC (voir 2.3.4).

Le système comporte un interpreteur de grammaire qui encode les informations sur la façon dont les bâtiments doivent être construits, et en partant d'empreintes au sol, génére semi-automatiquement un ou plusieurs bêtiments.

Les données de la base de règle pour cet interpreteur proviennent soit :
— Du module pour la génération d'un ensemble de données d'apparences fournies par l'utilisateur en compléments de données OpenStreetMap.
— Un module pour la création automatique des règles via les algorithmes génétiques.

Avec des modèles sémantiques, toutes les géométries sont associées à une signification sémantique (bâtiment, mur, fenêtre, ...), et chaque élément sémantique peut être décrit avec des attributs spécifiques. Une visualisation basée sur la sémantique améliore l'usabilité du modèle et facilite l'interopérabilité ainsi que le développement et l'utilisation d'applications pour le développement urbain.

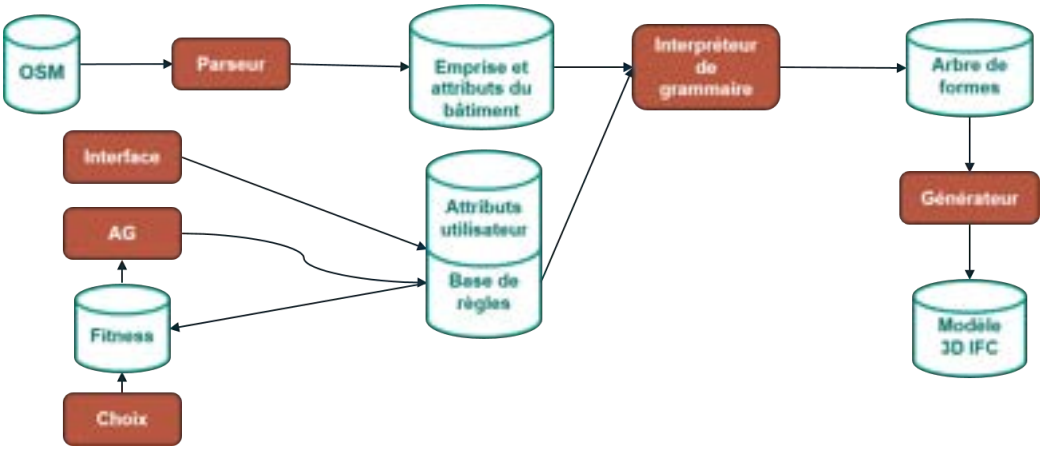

Figure 3.1 – Schéma du système

Le processus de génération pour le premier module, c'est à dire la base de règles, peut se dérouler de l'une de ces façons, l'utilisateur peut effectuer les actions suivantes :

— Il peut fournir un fichier de règles et un fichier contenant un ou plusieurs axiomes 3.3.1 (défini à la page 59) qui seront les emprises au sol.
— L'utilisateur peut également, via une interface graphique, créer un ensemble de données définissant les aspects généraux des bâtiments.
— Ou simplement fournir les coordonnées de la zone pour que le système puisse récupérér les données d'apparence correspondantes (si elles existent).

Le système va ensuite utiliser ces axiomes pour entamer le processus de génération, en interprétant les règles fournies successivement sur les emprises au sol fournies par l'utilisateur. Si des données d'apparence xml ont été fournies, celles-ci seront transformées en règles grammaticales avant d'être interprétées par le système.

Ces deux modules sont donc indépendants car l'utilisateur n'a pas besoin d'utiliser le jeu de données dans le cas où il souhaite générer manuellement des règles et les utiliser directement dans le deuxième module. Dans les sections suivantes, nous allons entrer dans plus de détails sur ces modules.

3.2 Généralités sur les grammaires

3.2.1 Grammaires de formes

Une grammaire de formes décrit des formes basiques subissant des transformations à travers des règles de remplacement pour produire des géométries structurées plus complexes. Elles ont été introduites par Stiny et Gips [40] pour créer des dessins dans des premiers travaux où toute la génération était manuelle. La forme la plus basique de ces grammaires consiste à utiliser des points, des lignes, des polygones, et des règles de transformation et de terminaison.

Une grammaire de formes est généralement composée de règles, de formes, et d'un moteur de génération qui en partant d'une forme initiale, sélectionne et traite les règles de manière récursive. Une règle définit la manière de remplacer les formes. Les règles sous-jacentes sont des transformations géométriques, c'est-à-dire mise à l'échelle, translation, rotation, etc. Si plusieurs règles s'appliquent, le moteur de génération sélectionne la règle à appliquer.

Une règle se compose de deux parties séparées par une flèche pointant de gauche à droite. La partie gauche de la flèche (LHS) représente une condition en termes de forme et de marqueur. La partie droite de la flèche (RHS) montre comment la forme LHS doit être transformée et où le marqueur est positionné. Le marqueur aide à localiser et à orienter la nouvelle forme.

L'aspect spatial des grammaires de formes est crucial pour sa mise en oeuvre dans la création et la conception des objets architecturaux.

Vu le développement des grammaires de formes utilisées pour l'analyse des objets architecturaux, ces dernières en tant qu'outil analytique ont un rôle non négligeable dans les recherches futures dans le domaine de l'architecture. L'expansion des technologies numériques et le perfectionnement de

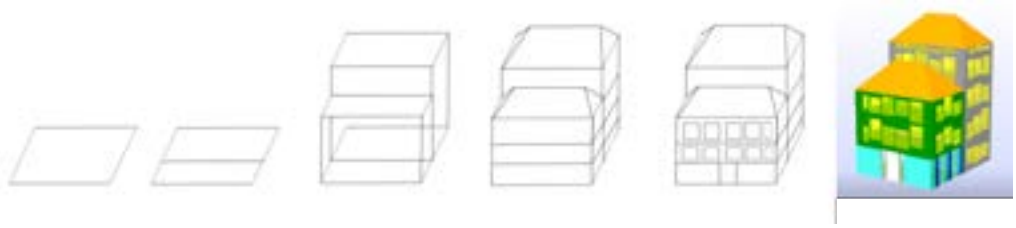

Figure 3.2 – Une grammaire de forme. Les règles montrent comment les formes doivent être transformées.

son utilisation pratique dans les études d'architecture fournissent une base nécessaire pour une nouvelle approche qui présente un fort potentiel.

Les grammaires de formes peuvent également aider à comprendre l'émergence et la structure des formes, leur composition visuelle.

3.2.2 Grammaires de formes appliquées à l'architecture

Les grammaires sont particulièrement adaptées pour encoder les connaissances sur l'architecture. Ce genre de système est en effet capable de générer des formes en gardant une certaine cohérence stylistique tout en offrant de possibles diversifications. Un ensemble de nombres finis de règles et de formes peut générer un nombre indéfini de solutions de conception. De plus, cela peut être utilisé comme outil d'analyse, pour la décomposition de formes complexes et comme outil de synthèse, générant des formes complexes à partir d'une forme simple.

3.3 L'interpréteur de grammaire

3.3.1 Interpréteur de grammaire

Nous proposons d'utiliser une grammaire de formes stochastique, en trois dimensions, sans contexte et avec attributs, pour la modélisation des bâtiments. La génération construit en premier lieu le modèle tridimensionnel, puis continue à structurer les façades et finalement divise la façade pour ajouter des détails pour les fenêtres, les portes, etc.

Une grammaire est un tuple composé d'un axiome, formes non terminales, formes terminales, et d'un ensemble de règles de réécriture et de probabilités associées à ces règles. L'interprète commence par analyser l'ensemble de règles en une carte des règles et en analysant l'axiome dans la première forme. À la fin du processus de modélisation, nous obtenons un arbre dont la racine est l'axiome (voir 3.3.1) avec une forme terminale à chaque feuille. Cet arbre d'analyse sera fourni sous forme d'entrée et les données obtenues pour construire le modèle IFC qui en résulte. Nous définirons quelques termes importants pertinents pour le système ci-dessous :

- **Formes :** Les formes sont les éléments constitutifs de la grammaire. Chaque forme se compose d'une géométrie, d'une liste d'attributs et d'un symbole provenant de l'ensemble de symboles non terminaux ou terminaux. Par exemple : un cube avec l'attribut *couleur : rouge* peut constituer un symbole non terminal si il doit subir d'autres transformations.
- **La géométrie** nécessite les éléments suivants :
  - L'empreinte de la forme : lors de l'exécution d'opérations en trois dimensions, la plupart des opérations nécessitent la modification de l'empreinte (redimensionnement, découpe, etc.), alors que pour une forme bi-dimensionnelle, le fonctionnement sur le point de départ et les dimensions sont suffisants pour la plupart des modifications.
  - Tel que définie dans [32], la portée est la zone de délimitation orientée, associée à la géométrie de la forme. Elle est définie par trois éléments : son point de départ qui est l'intersection des bords les plus à gauche et les bords les plus bas de l'empreinte, son système de coordonnées locale, et les trois dimensions de la boîte englobante.
- **L'axiome :** L'axiome est la forme initiale à partir de laquelle la construction commence. Pour notre grammaire, il s'agit d'empreintes au sol. Toutes les empreintes au sol de bâtiments peuvent être contenues dans un fichier texte, ou spécifiées à l'aide du format *shapefile*[1], qui est un format de fichier vectoriel très utilisé pour l'échange de données SIG. On peut voir un exemple de ce type de fichier dans la figure 3.3. Chaque empreinte au sol est un axiome qui correspondra à un

1. *Shapefile* est un format de données vectorielles géospatiales développé et réglementé par Esri pour les logiciels de système d'information géographique.

bâtiment. La dérivation commence par une extrusion verticale des bords. Les faces latérales, supérieures et inférieures résultant de l'extrusion sont étiquetées en conséquence. La grammaire sera réutilisée sur chaque polygone (axiome) contenu dans le fichier de forme qui permet d'éffectuer des opérations de transformation sur des empreintes de construction extraites des bases de données SIG.

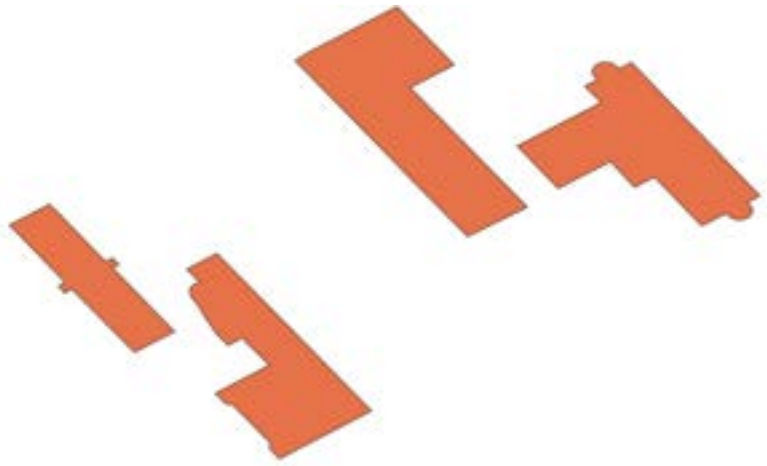

Figure 3.3 – Exemples de fichier axiome contenant les empreintes de quatre bâtiments.

— **Les règles de production :** Les règles sont structurées comme suit :

Structure d'une règle

```
predecesseur -> fonction( parametres ) {successeur1 : atr1
   = val1 , atr2 = val2 | successseur2} = prob
```

atr1 et *atr2* correspondent aux noms des attributs et *val1* et *val2* aux valeurs attribuées à ces attributs respectivement. La probabilité de sélectionner une règle est déterminée par la variable *prob*, et la somme des probabilités pour toutes les règles qui ont le même prédécesseur sur le côté gauche doit être égale à 1.

3.3.1.1 Éléments spécifiques aux bâtiments

La grammaire que l'on a développée ne repose pas sur des modèles externes, au lieu de cela chaque élément est directement construit en modèle

3D. La spécificité de notre grammaire se trouve au niveau des choix stylistiques dans la grammaire, combinés avec les règles de production, qui nous donnent suffisamment de flexibilité pour obtenir des modèles intéressants pour les applications envisagées. Un accent particulier a été mis sur les ouvertures, où plusieurs configurations possibles peuvent être obtenues à partir des attributs disponibles tels que illustrés à la figure 3.4. Par exemple, la fenêtre la plus à droite de la figure 3.4 est obtenue selon les règles ci-dessous :

Exemple de règles

```
yellowwindow -> rsplit (Z,0.2 ,0.3 ,0.3){wall : color=indianred
    | windowpart | upperwindowpart}
upperwindowpart -> rsplit(X,0.4,0.6){windowpart|windowpart}
windowpart -> {window: style=framed, framecolor=gold , material=
    glass}
```

La fenêtre est donc obtenue en découpant une forme initiale, non terminale (*yellowwindow*) en trois le long de l'axe Z : de bas en haut : *wall* qui a l'attribut couleur *indianred* , *windowpart* et *upperwindowpart*. La forme *upperwindowpart* est ensuite découpée suivant l'axe X en deux *windowpart* et les attributs de style, du cadre de la fenêtre (*framecolor*) et du materiel (*material*) sont définis.

Figure 3.4 – Exemples de configurations de fenêtres

La grammaire peut produire des modèles aussi spécifiques ou aussi génériques que nécessaire, sur la même base. L'utilisateur peut choisir d'utiliser certains éléments spécifiques de construction et la façon dont il en fait usage, ou il peut également utiliser des formes géométriques simples tout au long

du processus. Le style résultant du modèle variera selon ce choix. Cela peut être utile si l'utilisateur choisit plusieurs *shapefiles* pour différentes couches du terrain. Par exemple, il peut attribuer un symbole d'axiome par shapefile (par exemple : "lot" pour le shapefile avec des plans de construction, "grass" pour le shapefile avec la végétation) et ainsi lancer différentes dérivations sur les deux, pour mieux personnaliser le résultat.

Les terminaux : Nous utilisons quatre éléments terminaux : mur, fenêtre, balcon et porte. La plupart des choix stylistiques appliqués à ces terminaux, à l'exception des dimensions, sont gérés par les attributs tels que ceux détaillés ci-dessous.

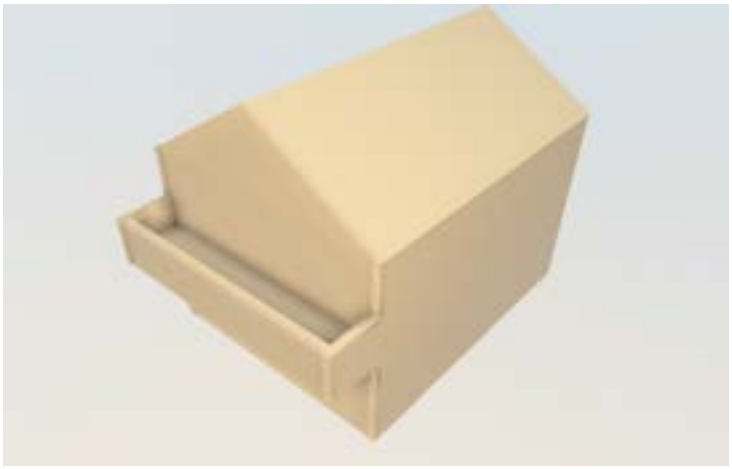

Figure 3.5 – Génération d'un balcon sur une façade de bâtiment

Attributs : chaque forme peut avoir un nombre arbitraire d'attributs définis par un nom et une valeur, précisant la façon dont les informations d'apparence seront traitées. Les attributs sont utilisés pour définir et propager ces informations stylistiques dans la hiérarchie des formes. Un attribut peut définir des informations telles que : style de toit, couleur des murs, si une ouverture est encadrée ou non (et si oui la couleur des cadres), ouverture opaque ou transparente (verre) et sa couleur, un balcon peut être introduit comme attribut sur une partie de façade, etc. La figure 3.6 illustre plusieurs bâtiments avec des fenêtres transparentes, murs blancs, portes et toîts rouges.

Figure 3.6 – Exemples de bâtiments

Toits : Nous avons inclus quelques formes de toit communes dans la grammaire, qui fonctionnent comme attributs de bâtiment. Le type de toit et la couleur sont spécifiés dans les attributs. L'utilisateur peut choisir parmi : plat, mansardé, à pignons, en appentis, en croupe.

Figure 3.7 – Types de toits

Comme il est expliqué par Laycock et Day [27], le toit mansardé est obtenu simplement en reliant la face supérieure du bâtiment avec le même polygone, réduit à 80% de sa taille. Pour les autres, le calcul du squelette droit du polygone est nécessaire. Le toit de hanche est obtenu directement en extrayant chaque face du toit du squelette droit. Nous extrayons les faces en trouvant le chemin le plus court entre les sommets de chaque bord du polygone initial (ce n'est pas le chemin direct). Pour ce calcul, nous rejetons n'importe quel autre bord du polygone initial et ne gardons que le squelette droit et le bord sur lequel nous opérons, comme le montre la figure 3.8,

puisque chaque face ne correspond qu'à un seul bord du-dit polygone.

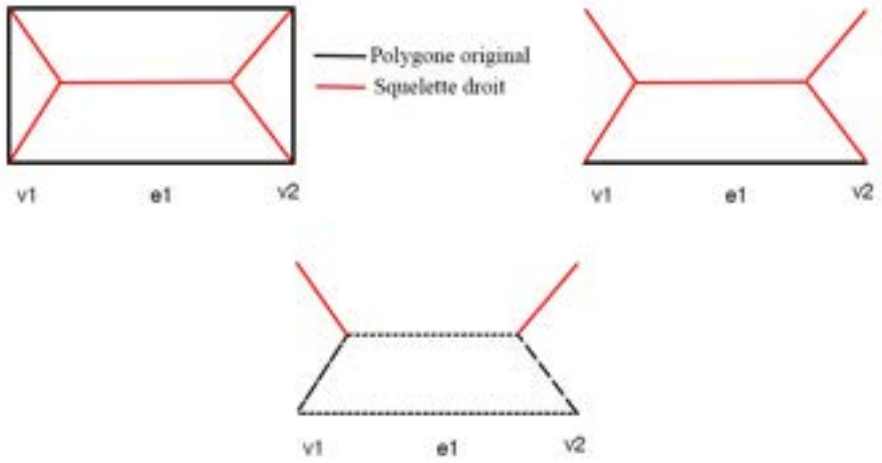

Figure 3.8 – Extraction des faces : les bords avec des lignes pointillées sur l'image en bas à droite forment la face extraite. Source : Nivolala et al. [34]

Pour le toit à pignons, nous partons du squelette droit formé par le polygone du toît et partant d'un point, créé par l'intersection de deux bissectrices (la droite qui coupe en deux parties égales l'angle formé à partir des coins du polygone), ce point est déplacé vers le point central de la ligne qui est incident aux deux bissectrices qui a créé le point d'intersection [27].

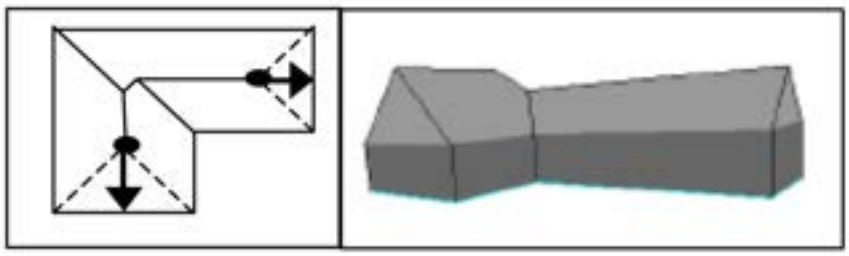

Figure 3.9 – A gauche : ajustement du toit à pignons, à droit : style de toit à pignons. Source : Laycock et Day [27]

3.3.1.2 Fonctions de transformation

Les fonctions suivantes peuvent être appliquées sur chaque forme pendant la dérivation :

— **Extrusion :** Effectue une extrusion verticale sur l'empreinte au sol, généralement appelée au début du processus de modélisation pour transformer un polygone 2D en forme 3D.

— **Décomposition :** Introduit par Müller et al. [32], cette fonction est utilisée pour décomposer une forme 3D en plusieurs composants de façades. Les formes passent de trois à deux dimensions afin de simplifier les opérations qui leur seront appliquées par la suite, cela se fait en ignorant un axe en définissant la dimension de la forme suivant cet axe à zéro. L'axe X est calculé suivant le bord inférieur du polygone, définissant le référentiel le long de la boîte englobante.
Ces façades peuvent être sélectionnées avec les sélecteurs suivants :

1. façade avant, arrière, gauche, droite définis par les mots clés *front, back, right, left*
2. *side* pour séléctionner pour toutes les façades.
3. *others* pour les façades non encore sélectionnées, par exemple : dans le cas où la façade avant a été séléctionnée auparavant, *others* selectionnera les façades arrière, gauche et droite.

Les côtés avant, arrière, sont définis en référence à la forme parent, si aucune forme parente n'est présente, alors ils sont définis en référence à un pivot (X,Y,Z) où Y pointe vers l'arrière , -Y à l'avant, X pointe vers la droite et -X pointe vers la gauche, selon le repère associé à la scène. Dans un premier temps, chaque face est étiquetée comme une face latérale. Nous considérons l'axe X de la forme parente et son centroïde et calculons le produit scalaire de point du-dit axe et l'axe de la forme actuelle pour définir l'orientation.

— **Découpe :** Coupe une forme en plusieurs autres formes le long de l'axe sélectionné et avec les dimensions spécifiées dans les paramètres. Dans la figure 3.10, la forme initiale du bâtiment a été découpée en quatre étages suivant l'axe Z avant d'être décomposées en pièces.

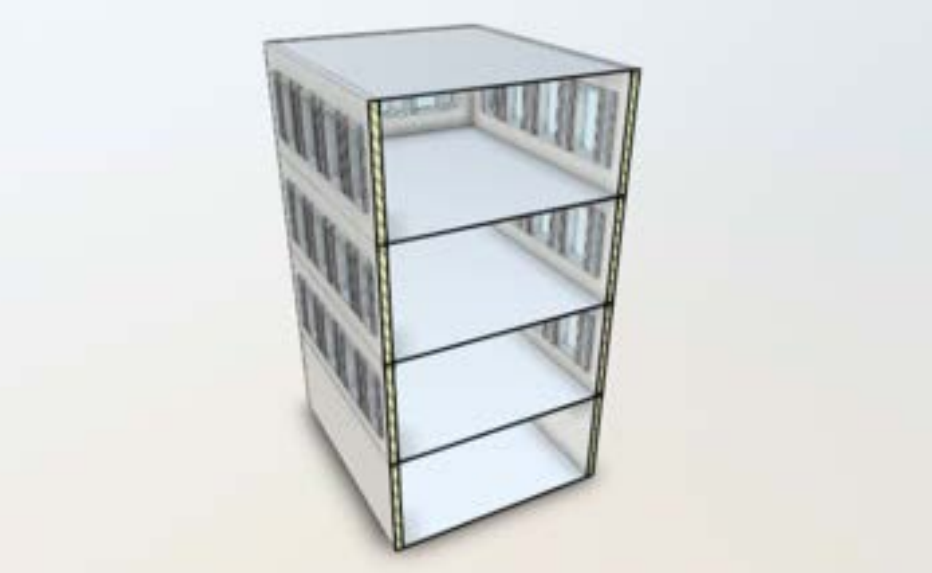

Figure 3.10 – Découpe d'un bâtiment en quatre étages

— **Découpe relative :** La même opération que la division avec les valeurs relatives à la forme parente.
— **Répétition :** Répète les successeurs le long de la forme parente pour l'axe et les dimensions spécifiés, tant qu'il y a encore de l'espace libre sur la forme parente.
— **Translation :** Traduit la forme en déplaçant le point de départ et les points de l'empreinte
— **Rotation :** Fait pivoter les axes et les points de l'empreinte
— **Redimensionnement** : Redimensionne la forme et déplace les points de l'empreinte.

3.3.1.3 L'arbre de formes

Les formes sont structurées sous forme d'arbre et classées suivant leur symbole et leurs balises qui indiquent leur place dans la hiérarchie spatiale. Il existe six types d'étiquettes : bâtiment, partie de bâtiment, étage, pièce, façade et partie de façade. Les modèles de masse tridimensionnels sont étiquetés en tant que bâtiments, sauf s'il y a indication du contraire lors de la lecture du fichier contenant les axiomes, auquel cas ils peuvent être traités comme d'autres éléments urbains tels que de la végétation. Toute subdivision

d'un élément étiqueté *bâtiment* (à l'exclusion d'une opération de décomposition) crée soit des *parties de bâtiments* dans le cas d'une coupe verticale ou des *étages* dans le cas d'une coupe horizontale, tant que les formes résultantes sont également tridimensionnelles. Une opération de découpe sur un étage créera des *pièces*, enfin la fonction de décomposition lie l'étiquette de façade aux formes successeurs ; tous les éléments résultant des subdivisions sur ces façades sont également étiquetés en tant que *partie de façade*.

Ci-desssous est un exemple d'arbre de formes.

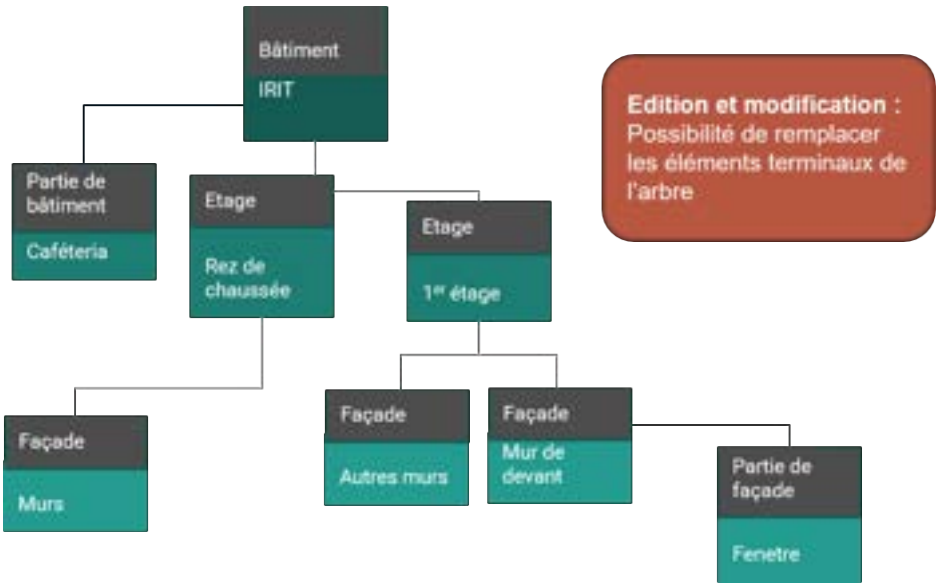

Figure 3.11 – Exemple d'arbre de formes

Modifications sur l'arbre de formes : Une fois qu'un premier arbre a été obtenu, il est possible pour l'utilisateur de le découper au niveau d'un symbole pour arrêter la dérivation à ce niveau (et remplacer automatiquement ce symbole par un terminal), ou de rajouter une nouvelle règle pour un prédécesseur : ajouter une découpe comme règle possible sur un mur par exemple.

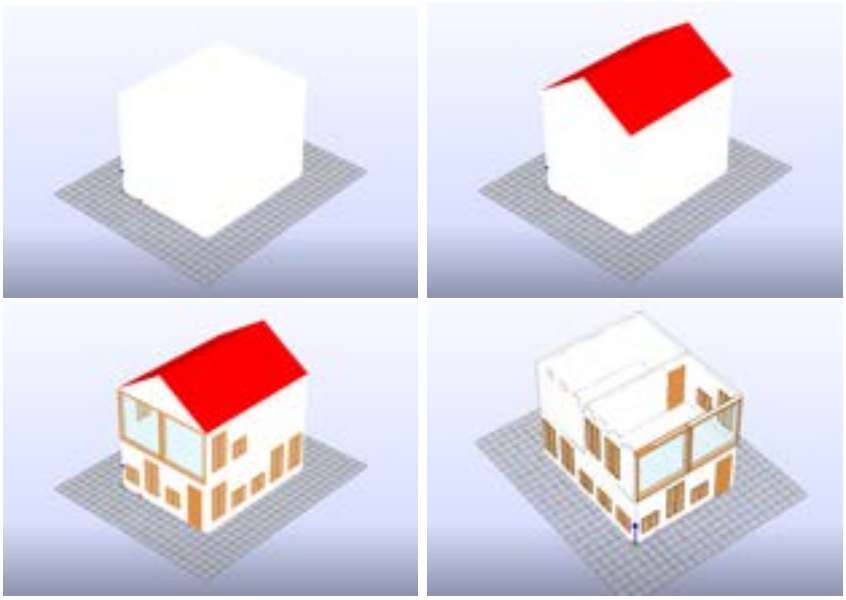

Figure 3.12 – Différents niveaux de détails obtenus avec notre système

Niveau de détails (LOD) [2] Le niveau de détails nécessaire peut être passé comme argument pendant le processus de génération. Nous pouvons générer quatre niveaux de détails différents, qui sont obtenus en récupérant et en modélisant un sous-ensemble de l'arbre de formes pour chaque niveau de détail, et en allant plus profondément si plus de détails sont nécessaires : les portes et fenêtres, par exemple, sont prises en compte à partir du LO3.

Pour la réalisation de chaque LOD (de 1 à 4) différent, le module de génération traverse l'arbre jusqu'à un point dépendant du niveau souhaité. Pour LOD1, le module ne cherche pas plus bas que les formes marquées comme *bâtiment*, LOD2 ajoute l'élément de *toit* dans la génération, LOD3 génère l'extérieur des bâtiments : étages formes, LOD4 considère chaque feuille de l'arbre de formes, toits, exterieurs et intérieurs inclus.

3.4 La base de connaissances

Pour faciliter le processus et réduire le temps requis pour la génération de règles, une interface est créée permettant à l'utilisateur de pouvoir se référer

2. Level of Details

à une base de connaissances. Il peut créer cet ensemble de données, composé de zones géographiques organisées hiérarchiquement avec leurs informations correspondantes, ainsi que des jeux de règles, et le remplir tout au long, avec de nouvelles zones, et/ou des jeux de règles à partir d'une interface graphique.

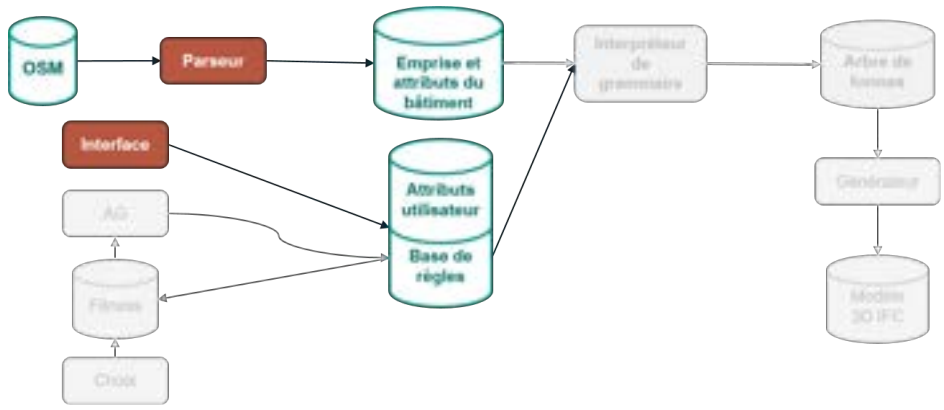

Figure 3.13 – Module pour le remplissage de la base de règles

Lorsque nous utilisons des fichiers au format OpenStreetMap, nous avons la possibilité de récupérer automatiquement les règles correspondantes (en fonction de l'emplacement) à partir de cette base de connaissances, où les informations d'apparence sont organisées dans la hiérarchie, de l'entité la plus large qui comprend tous les autres emplacements : la terre, jusqu'aux informations typiques d'une certaine zone, comme un quartier. En effet, la base de connaissance contient une ensemble de données qui sont des informations et attributs fournis par l'utilisateur, utilisés conjointement avec les informations et attributs qui peuvent être extrait d'OSM pour des zones organisées de manière hierarchique. Les informations d'apparence se propageront de la plus grande à la plus petite zone par défaut, par exemple si le nombre d'étage par défaut pour Toulouse n'est pas spécifié, on se référera aux nombre d'étage pour la région Midi-Pyrennées. Les informations de la plus petite zone seront récupérés pour des bâtiments donnés.

Figure 3.14 – Informations d'apparence organisées hiérarchiquement

3.4.1 L'interface graphique

Une interface graphique est fournie pour faciliter la construction du jeu de données, ainsi que la définition de règles plus génériques. Via cette interface, les informations d'apparence ainsi que certaines informations géométriques de base peuvent être fournies par l'utilisateur. Sans rentrer dans les détails d'implémentation, les bâtiments générés suivent un modèle générique, mais un certain nombre de paramètres peuvent être modifiés pour donner des résultats personnalisés, ouvrant ainsi la possibilité d'un nombre considérable de variations. Le processus commence par la sélection ou la création de la zone que l'utilisateur souhaite remplir (dans le volet gauche de l'interface), en fournissant les coordonnées et le rayon de la zone. Une fois cela fait, en sélectionnant "éditer les règles" il a la possibilité de sélectionner des paramètres géométriques et esthétiques, tels que les couleurs du bâtiment, la hauteur minimale et maximale d'un bâtiment, la position des ouvertures, etc. Plusieurs informations d'apparence tels que la hauteur des bâtiments, la couleur, le type de toit, la taille des fenêtres, etc. peuvent être ajoutées par l'utilisateur au même endroit avec des probabilités variables définies pour les éléments du bâtiment.

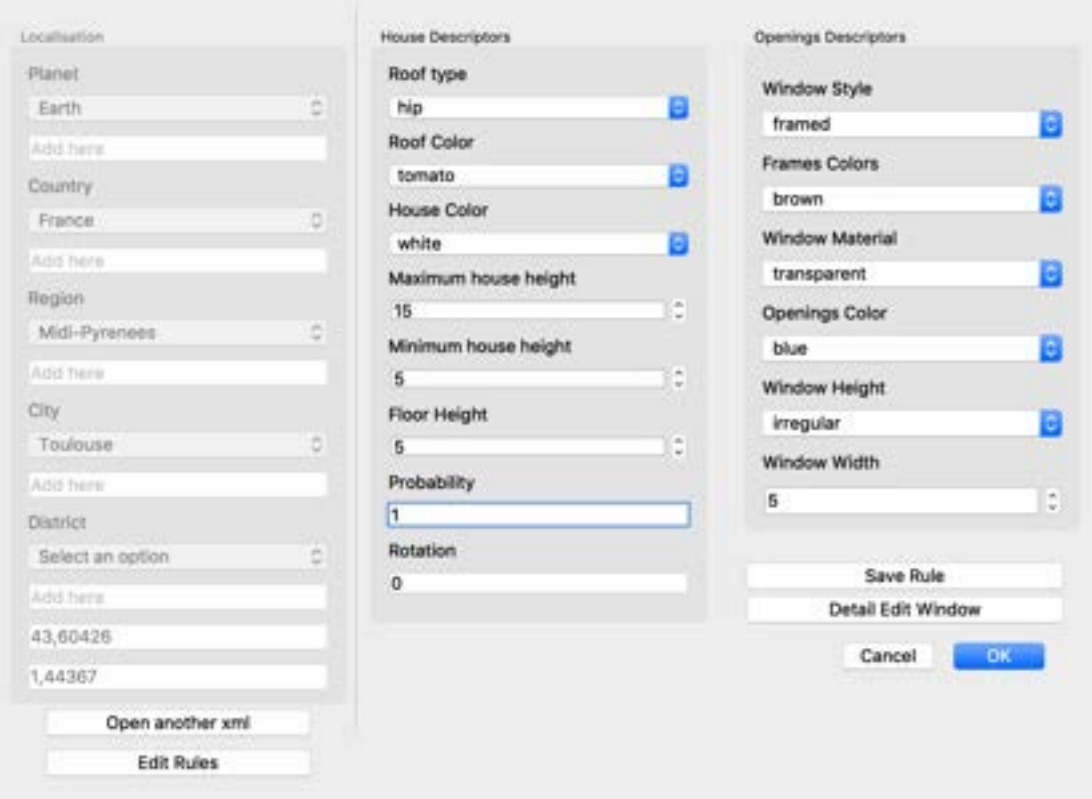

Figure 3.15 – L'interface graphique

L'utilisateur peut modifier les ouvertures avec plus de contrôle, car elles peuvent être dessinées séparément dans une autre interface (accessible en sélectionnant dans la fenêtre "Detail Edit Window") qui ouvre un panneau représentant l'ouverture et permettant de dessiner des lignes et de subdiviser en parties rectangulaires. Chaque division est traduite en une règle de découpe, les paramètres sont calculés avec la position de chaque ligne placée par l'utilisateur relativement à toute la largeur ou la hauteur du rectangle parent. Ces fractions sont fournies comme arguments à une règle de découpe relative. Toutes ces informations seront stockées dans le jeu de données et peuvent ensuite être récupérées et traduites en jeux de règles.

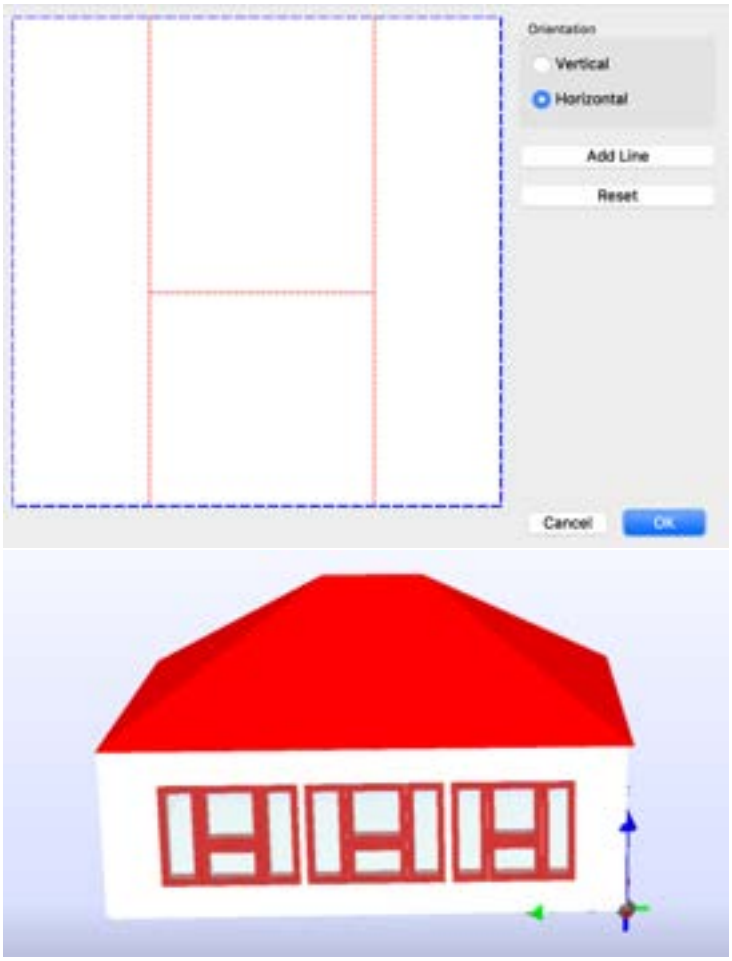

Figure 3.16 – Édition détaillée d'une ouverture et résultat

3.4.2 Exemple

L'exemple illustré ci-dessous a été généré avec cet interface. Sur la zone du milieu, les bâtiments ont quatre étages avec des façades en gris, le reste est en brique sur un seul étage. Pour séparer les différents bâtiments qui doivent correspondre à différentes règles, le système a pris en entrée deux différents *shapefiles*, donnant au système deux couches correspondant à deux axiomes sur lesquelles les règles sont appliquées.

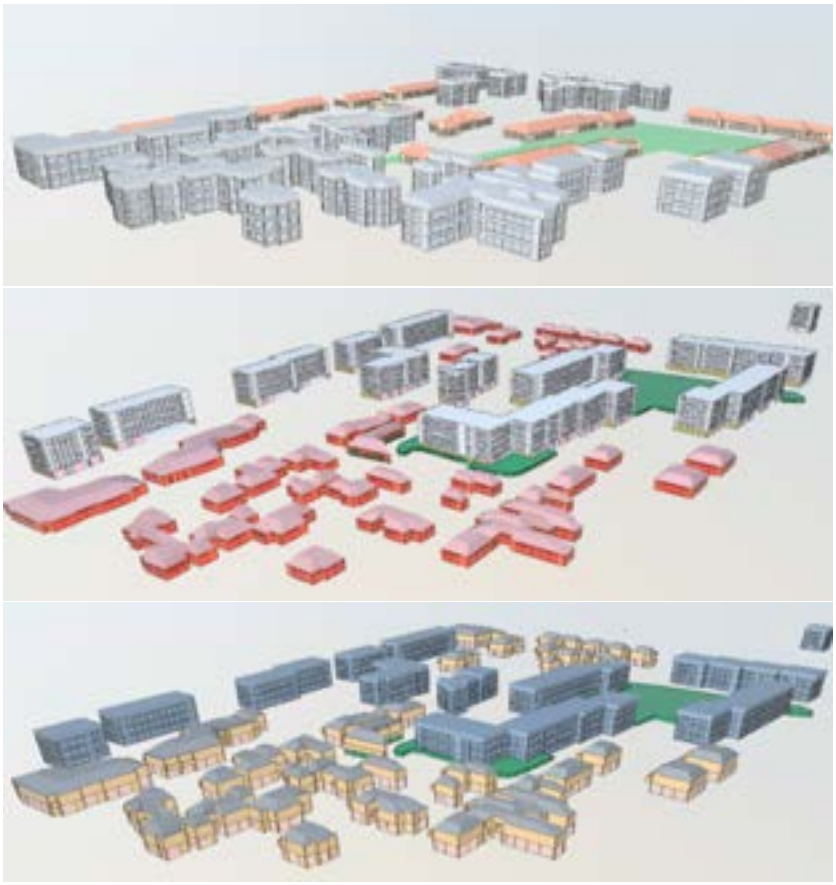

Figure 3.17 – Exemple de génération d'une Même zone avec des styles architecturaux différents

La séparation entre les bâtiments aurait également pu se faire en définissant dans le fichier de données XML les zones ggéographiques sous formes de polygones (au lieu de cercles) correspondant à un changement général d'apparence pour les bbâtiments environnants.

3.5 Export vers IFC

3.5.1 Conception de composants sémantiques

Alors que le processus de modélisation crée des formes sans sémantique, nous allons utiliser l'arbre d'analyse pour construire des éléments architec-

turaux majeurs, tels que des pièces, des toits, des portes ou des fenêtres. Il est possible d'extraire des informations sémantiques à partir de la géométrie, par le biais des fonctions utilisées sur les formes ainsi que de leurs symboles terminaux. Pour cette partie, nous avons fait le choix d'utiliser le format de fichier IFC (détaillé à la section 2.3.4), un modèle de construction unifié pour conserver les informations sémantiques. IFC est un format de fichier normalisé qui représente une spécification pour les données de modélisation de l'information de bâtiment (BIM) qui sont échangées et partagées entre différentes personnes impliquées dans un processus de construction de bâtiment ou dans la gestion de bâtiment.

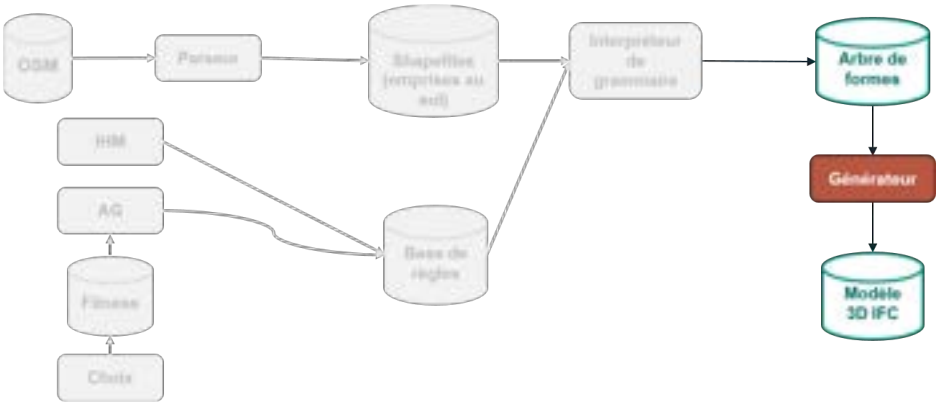

Figure 3.18 – L'export IFC

IFC est la norme internationale de l'openBIM. IFC a été choisi pour l'export car c'est l'un des standards les plus expressifs entièrement centré sur la modélisation des bâtiments. C'est un modèle complet qui intègre la spatio-cohérence sémantique, particulièrement utile dans la construction de bâtiments, mais aussi l'urbanisme, et il peut facilement être converti en d'autres formats plus compacts si nécessaire, cependant, il n'y a pas beaucoup de moteurs procéduraux faisant le lien vers ce modèle.

Un modèle IFC contient les données de géométrie et de construction du bâtiment. Le fait que le modèle contienne des données géométriques et non géométriques sur le projet de construction le rend extrêmement complet. Il définit également les relations entre les éléments de construction, comment ils sont connectés, structurés dans l'espace ou regroupés. Un modèle de zone

(IfcZone) est une aggregation d'espaces organisés comme un ensemble hiérarchique d'instances de classe IFC : le processus commence par un projet qui contient toutes les zones, une zone contient des bâtiments donnés par leur empreinte, un bâtiment est composé de plusieurs étages, un étage contient des pièces, des murs, etc et enfin les murs peuvent avoir des ouvertures qui contiennent des portes, fenêtres, éléments décoratifs, etc.

3.5.2 De l'arbre de formes à la hiérarchie IFC

Afin de respecter la façon dont les éléments sont construits, suivant une conception architecturale standard en IFC, une étiquette a été attachée à chaque forme après toute opération pour simplifier la transition comme nous l'avons vu auparavant (à la section 3.3.1.3). Étant donné que les hiérarchies sont différentes entre l'arborescence résultante et le modèle requis par le format IFC, certains indicateurs doivent être fixés aux formes de l'arbre de formes. Nous allons décrire la génération d'un bâtiment entièrement détaillé ci-dessous : le processus de génération consiste à traverser les niveaux de l'arbre de formes et à générer un élément IFC pour chaque élément pertinent qui doit être instancié.

Dans un premier temps un élément est créé pour le site dans sa globalité, ensuite le moteur récupère tous les éléments marqués partie de bâtiment qui ne contiennent pas directement de partie de bâtiment eux-mêmes, mais peut contenir des étages (qui à leur tour peuvent contenir des parties de bâtiments ou pas). Chaque bâtiment obtient un IfcPlacement pour le situer par rapport au site, est stocké sous forme d'IfcBuilding et lié au site. Pour toutes les parties du bâtiment, les éléments fils marqués comme étant un étage sont récupérés et une forme de plafond est générée à partir de son empreinte. Le toit est construit pour l'étage le plus élevé, et les cloisons intérieures avec la géométrie correspondante est créée. Pour chaque mur de chaque étage est créé un IfcWallStandardCase : une entité IFC qui contiendra la géométrie des murs, et il lui est donné un placement par rapport à sa forme mère qui est l'étage, associé à une représentation solide, ainsi qu'une représentation d'axe définie par les deux extrémités du mur. La construction d'éléments solides commence vraiment avec les murs. De la géométrie de chaque forme de l'arbre de formes est extrait une liste de points : le point de départ, la taille et la direction nous donnent l'ensemble des points nécessaires pour créer une projection au sol. Ce sont les données nécessaires pour la création d'un IfcEx-

trudedAreaSolid : un solide défini par l'extrusion d'une zone 2D étant donné une direction et une profondeur. Une fois cette entité définie, la géométrie du mur est créée. Les toits sont créés en récupérant directement les coordonnées des faces, calculées par les différents algorithmes expliqués précédemment, et en instanciant un IfcFacetedBrep (qui est une forme simple de modèle de représentation où toutes les faces sont planes et toutes les arêtes sont des lignes droites) au-dessus des étages les plus élevés pour chaque partie du bâtiment. Enfin, la création d'ouvertures est essentiellement la même que ci-dessus, mais d'abord pour créer le trou, un IfcRelVoidsElement est créé entre le mur et l'élément d'ouverture, créant une relation de soustraction de la forme dans le mur où l'ouverture se situera, avant d'extruder l'ouverture réelle au même endroit.

L'export du fichier IFC se déroule comme suit :

— Toutes les pièces de construction instancient IfcBuildings : IfcBuilding est la classe utilisée pour construire et contenir la structure spatiale d'un bâtiment.
— Ces bâtiments contiennent des entités IfcStorey définis par les éléments étiquetés étages, et les toits. Un IfcStorey définit un étage associé à un bâtiment. Il peut s'étendre sur plusieurs étages reliés et peut également être décomposé en parties (horizontales), où chaque partie définit un étage partiel.
— Les éléments étiquetés façade sont utilisés pour instancier les murs pour chaque étage.
— Les ouvertures sont créées là où les portes et les fenêtres sont censées être en faisant un trou de la forme prévue sur les murs, avant d'y insérer les formes réelles.
— S'il y a d'autres divisions sur une façade qui n'est pas une ouverture, chaque partie de façade instancie un IfcCurtainWall : un mur extérieur d'un bâtiment qui est un assemblage de composants, suspendu au bord de la structure plutôt que de s'appuyant sur le sol.

3.6 Implémentation et résultats

3.6.1 Un exemple de bâtiment

Cette section introduit le processus de modélisation tel qu'il se déroule avec notre système. Pour obtenir le résultat tel qu'il est montré dans la figure 3.19, et suivi par le jeu de règles correspondant, le modèle de masse initial est divisé en deux bâtiments (b1 et b2) avec des styles de toit de hanche. Le premier bâtiment est b1, le deuxième bâtiment b2 est d'une taille plus réduite, obtenue grâce à une valeur aléatoire et devient la forme nommée b3, puis les deux bâtiments sont divisés en cinq étages de même hauteur chacun, avec l'opération de subdivision (ils finissent avec respectivement deux et quatre étages pour respecter les hauteurs définies précédemment, même si les paramètres de divisions fournies par l'utilisateur sont incohérentes), le rez-de-chaussée (gfloor) de b3 est défini à part pour être différent de tous les autres étages. Chaque étage est divisé en façades qui contiennent une répétition de fenêtres en verre jaune. Les hauteurs de la fenêtre jaune sont définies comme aléatoires dans une fourchette de valeurs définies.

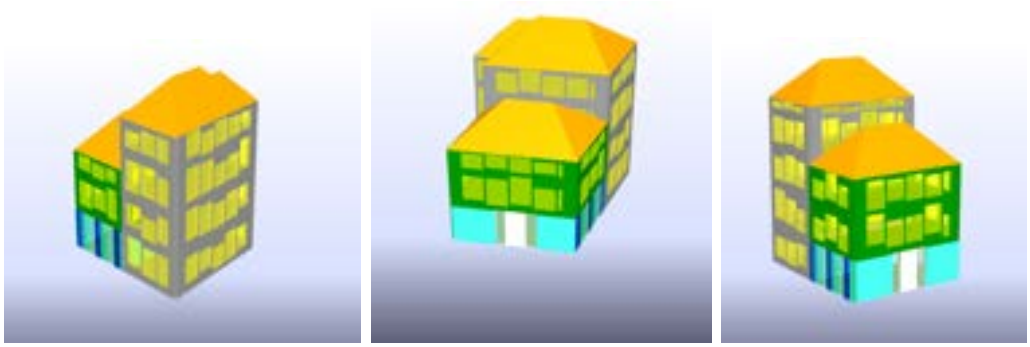

Figure 3.19 – Exemple de bâtiment

Règles pour la figure ci-dessus

```
lot -> extrude(20){building:roof=hip,roofcolor=sandybrown}
building -> split(Y,sy/2,sy/2){b1|b2:color=grey}
b1 -> S(sx, sy, sz*rand(0.6,0.8)){b3:color=green}
b2 -> repeat(Z,4){floor}
b3 -> split(Z,4,4,4,4,4){gfloor|floor|floor|floor|floor}
floor -> comp(){side:face}
```

```
gfloor -> comp(){left:gleft_facade|front:gfront|right:
    gright_facade|back:gback}
face -> rsplit(X,0.1,0.8,0.1){wall|windowarea|wall}
windowarea -> repeat(X,2,0.2){vwindow|wall}
vwindow -> rsplit(Z,rand(0.1,0.3),rand(0.5,0.8),0.5){wall|window
    :color=yellow,material=glass|wall}
gfront -> split(X,5,2,5){wall:color=turquoise|door:color=white|
    wall:color=turquoise}=0.4
gfront -> split(X,4,1,2,1,4){wall:color=turquoise|window:color=
    pink,material=glass|door:color=white|window:color=pink,
    material=glass|wall:color=turquoise}=0.6
gright_facade -> repeat(X,1,1,1){window:color=blue|window:color=
    turquoise,material=glass|window:color=lightblue,material=
    glass}
gleft_facade -> repeat(X,1,1,1){window:color=blue|window:color=
    turquoise,material=glass|window:color=lightblue,material=
    glass}
gback -> wall
```

3.6.2 Exemple avec le jeu de données

Voici un exemple qui a été généré à partir de l'interface utilisateur, sans écrire manuellement le jeu de règles. Les ouvertures ont été dessinées via l'interface. L'extrait suivant montrant l'ensemble de données correspond à l'un des deux styles de maison illustrés dans la figure 3.20, les informations d'apparence peuvent être trouvées comme attributs à l'élément *values*, l'apparence de la fenêtre est codée dans la balise *opening* :

Extrait de la base de connaissance

```
...
<town designation="Antananarivo">
<location lat="-18,9333" lon=47,5167"" dist="0,3"/>
<appearance>
<values proba="0,5" bdRooftype="gabled" color="peru" doorCol="
    ivory" floorsize="5" maxHeight="20" maxrotation="0" minHeight
    ="5" openingCol="paleturquoise" roofColor="darkred"
    windowfcolor="ivory" windowheight="tall" windowmaterial="
    glass" windowstyle="framed" windowwidth="2" />
<opening>
<split axis="X" children="nt4582467848|nt4582467288" parent="
    nt4582467064" splitvalues
    ="0.5023809523809524,0.4976190476190476" />
```

```
<split axis="Z" children="nt4582467512|nt4582468296" parent="
    nt4582467848" splitvalues
    ="0.4595238095238095,0.5404761904761906" />
<split axis="Z" children="nt4582467736|nt4582468856" parent="
    nt4582467288" splitvalues
    ="0.4595238095238095,0.5404761904761906" />
<elements parent="nt4582467064" terms="nt4582468856,nt4582467736
    ,nt4582467512,nt4582468296" />
</opening>
</appearance>
...
```

Figure 3.20 – Exemple généré avec le jeu de données

3.6.3 Exemples avec des bâtiments réels

Pour les résultats suivants, nous avons créé nos jeux de règles basés sur des bâtiments du monde réel, ces jeux de règles ont été réutilisées sur plusieurs empreintes des zone géographiques concernée. Dans ce cas, le centre-ville de Toulouse (figure 3.21) et les bâtiments d'une université (figure 3.22). Leurs empreintes au sol ont été extraites d'OpenStreetMap, et nous avons généré l'herbe à partir d'un fichier *shapefile* séparé via l'extrusion verticale de l'axiome représentant les zones où il y a l'herbe. Les bâtiments universitaires ont 87145 polygones qui ont été générés en moins de 142,03 secondes, le centre-ville a 119448 polygones et la génération a duré 200,1 secondes. Nous avons effectué nos tests sur un processeur Intel(R) Core i5-4690 @ 3,50 GHz avec 8Go RAM.

Toulouse Downtown ruleset

```
lot -> extrude(15){building:color=blanchedalmond,roof=hip,
    roofcolor=darksalmon}=0.4
lot -> extrude(20){building:color=silver,roof=hip,roofcolor=
    darksalmon}=0.2
lot -> extrude(15){building2:color=lightsalmon,roof=hip,
    roofcolor=darksalmon}=0.2
lot -> extrude(20){building:color=palegoldenrod,roof=hip,
    roofcolor=lightsalmon}=0.2
building -> repeat(Z,5){floor}
building2 -> repeat(Z,5){floor}
floor -> comp(){side:face}
floor2 -> comp(){side:face2}
face -> rsplit(Z,0.9,0.1){upper_face|wall:color=darksalmon}
face2 -> rsplit(Z,0.9,0.1){upper_face2|wall:color=white}
upper_face -> rsplit(X,0.1,0.8,0.1){wall|windowarea|wall}
upper_face2 -> rsplit(X,0.1,0.1,0.8,0.1,0.1){wall:color=
    darksalmon|wall|windowarea2|wall|wall:color=darksalmon}
windowarea -> repeat(X,2,1){vwindow|wall}
windowarea2 -> repeat(X,2,1){vwindow2|wall}
vwindow -> rsplit(Z,0.25,0.5,0.25){wall|bwindow|wall}
vwindow2 -> rsplit(Z,0.2,0.6,0.2){wall|bwindow2|wall}
bwindow -> rsplit(Z,0.1,0.8,0.1){wall:color=darksalmon|b_window|
    wall:color=darksalmon}
bwindow2 -> rsplit(Z,0.3,0.7){wall:color=white|window:color=
    lightgrey}
b_window -> rsplit(X,0.1,0.8,0.1){wall:color=darksalmon|window|
    wall:color=darksalmon}
```

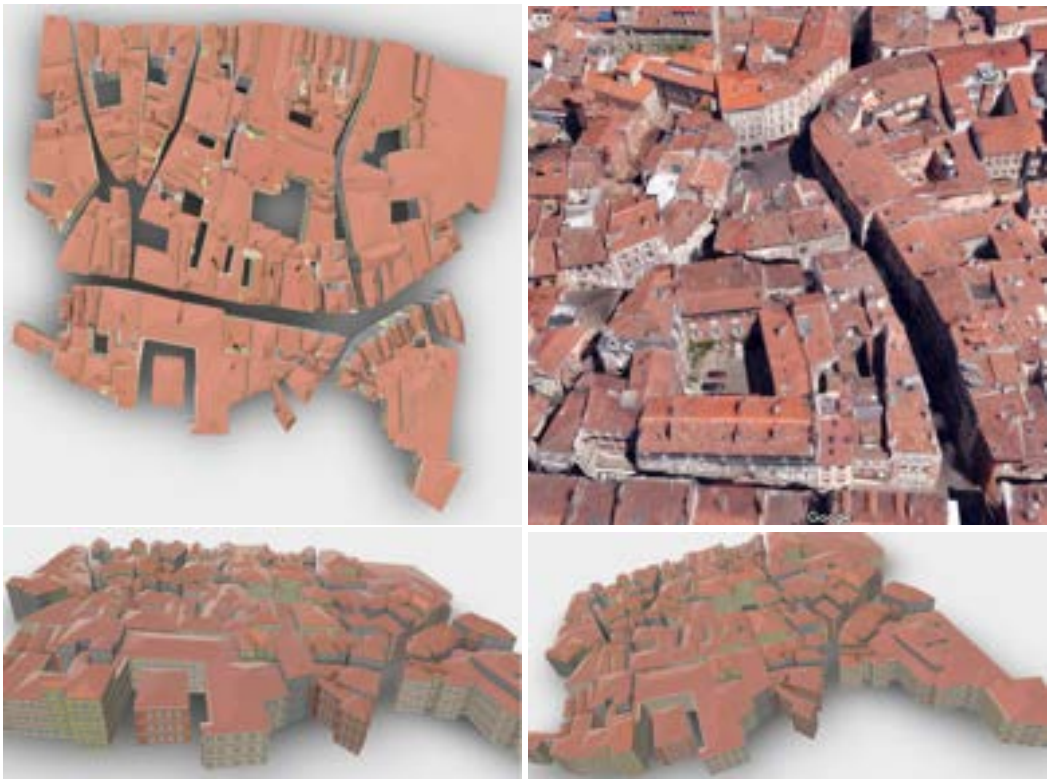

Figure 3.21 – Immeubles en centre ville (Image Google Maps comme référence en haut à droite)

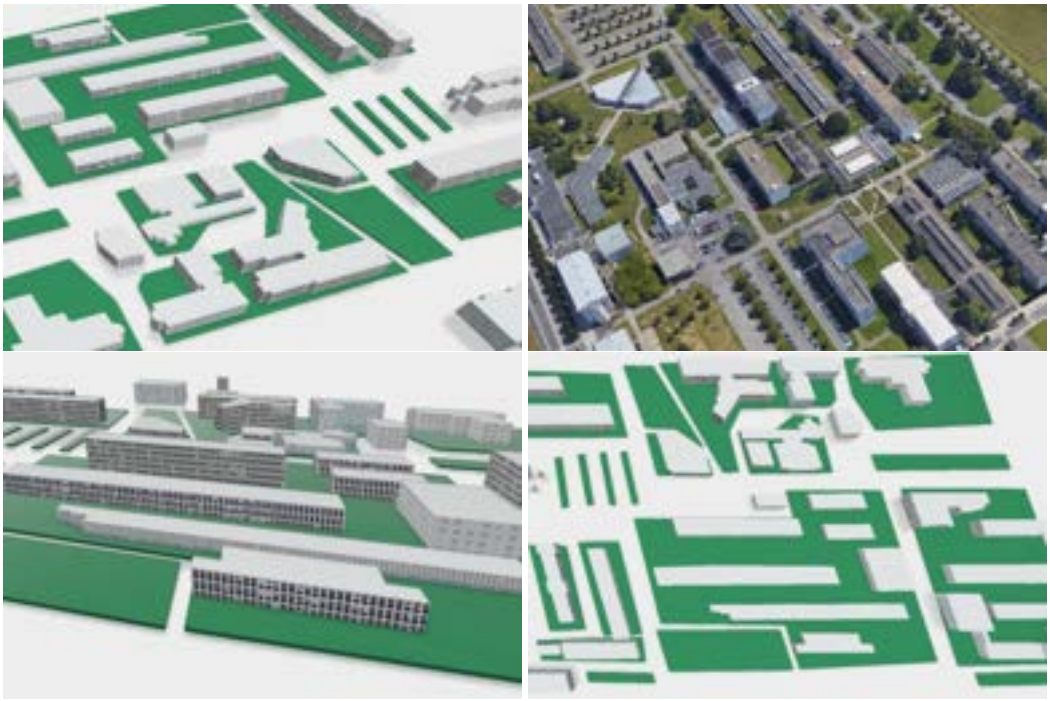

Figure 3.22 – Les bâtiments de l'Université (Image Google Maps comme référence en haut à droite

Dans ce qui suit, le bâtiment de l'Institut de Recherche en Informatique de Toulouse (figure 3.24) a été obtenu en prédécoupant l'emprise au sol provenant d'OSM en deux différentes parts du bâtiment. Deux règles différentes (dont notamment la forme des fenêtres) ont ensuité été générées via l'interface graphique pour les deux parts distinctes du bâtiment : la partie à quatre étages et la partie à un seul étage. Il est suivi du bâtiment U4 (figure 3.25) qui, initialement avec un style différent comme sur la photo, a été généré avec les mêmes règles que pour l'IRIT.

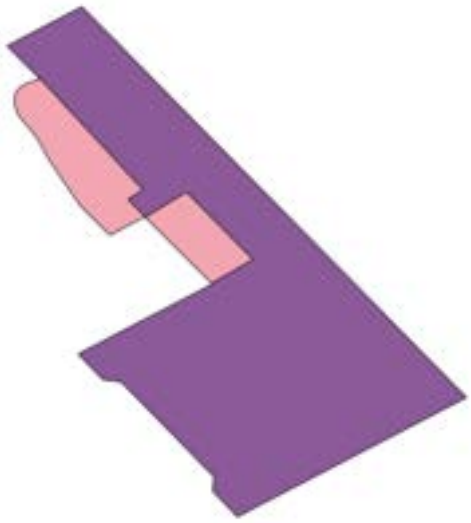

Figure 3.23 – Prédécoupage du fichier shapefile de l'IRIT

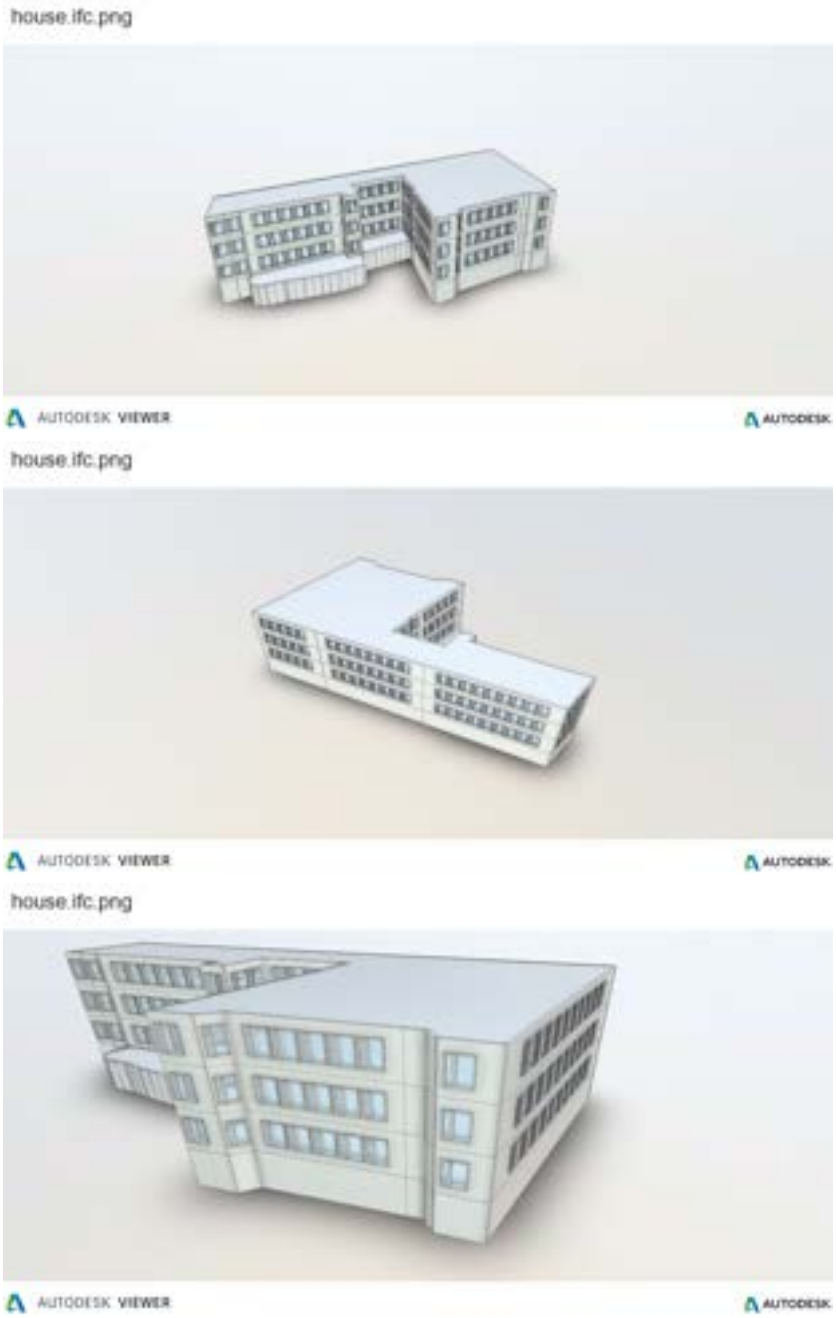

Figure 3.24 – IRIT

Figure 3.25 – Le bâtiment U4 généré avec les mêmes règles que celles del'IRIT

Enfin dans les figures qui suivent (figure 3.26), on a généré des maisons typiques malgaches, les apparences sont imités des maisons réelles dans la photo d'en haut. Puis sur la même empreinte, les maisons de la dernière photo du bas on été générés avec differents types de volets typique du centre ville, déssinés via l'interface, les informations de couleur ont egalement été fournies par l'interface.

Figure 3.26 – Maisons malgaches

3.7 Génération automatique de règles

Bien qu'expressive, il est néanmoins clair que la construction manuelle d'un ensemble de règles grammaticales reste une tâche non triviale. L'introduction d'un module pour l'automatisation de l'écriture de ces règles s'avère utile, et c'est ici que l'on a introduit les algorithmes génétiques qui offrent une technique de recherche et d'optimisation aléatoire guidée par le principe des systèmes génétiques naturels. Des travaux sur concept de transformation de grammaires ont été effectués sur les téléphones auparavant [23], mais nous nous rendons compte que quelques uns de ces concepts peuvent s'adapter à nos besoins.

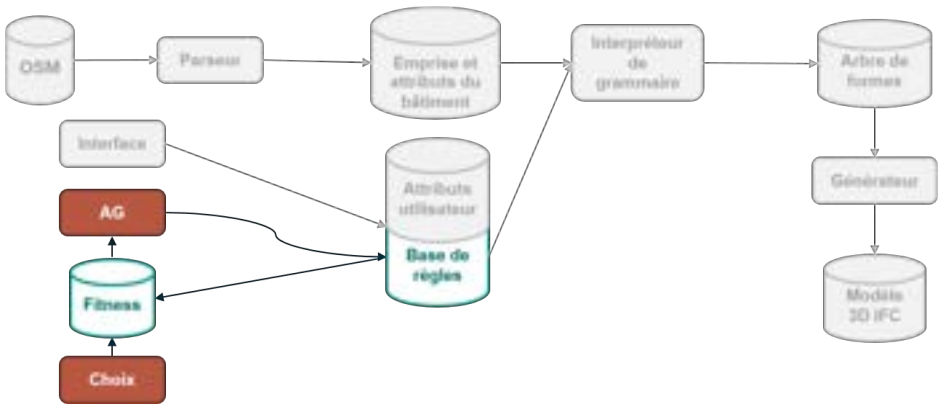

Figure 3.27 – Module pour l'automatisation de l'écriture de ces règles

Le choix des algorithmes génétiques permettra de trouver une solution optimale sans avoir à spécifier les étapes de la résolution du problème. Ces derniers se prêtent très bien aux problèmes d'optimisation. Le plus gros du travail consiste à fournir les objectifs et définir leur évaluation. Dans l'architecture en particulier, il est possible d'obtenir une définition assez claire et exhaustive de tels objectifs, esthétiquement parlant.

L'adaptation passe donc par la définition de deux contraintes : la première est l'esthétique reliée à l'architecture, et la seconde : les contraintes globales qui sont renforcées dans un souci de cohérence (cohérence des dimensions, du nombre de subdivisions sur une forme, ...).

3.7.1 Généralités sur les algorithmes génétiques

Les algorithmes génétiques sont un type de calcul évolutionnaire conçu par Holland [22] se basant sur la théorie de Darwin selon laquelle, durant le processus d'évolution et de sélection naturelle, les gènes qui survivent pour une population donnée sont ceux qui sont les plus adaptés aux besoins de cette population, résultant en des populations vivantes améliorées. Ces algorithmes sont appliqués à des individus qui sont les solutions potentielles à un problème et permettent d'obtenir une solution optimale à un problème où il n'y a pas de méthode exacte de résolution, l'environnement procurant une mesure d'évaluation permettant d'identifier les meilleurs individus.

Le processus est réitéré plusieurs fois, et de la même manière que pour l'évolution, les résultats ne seront visibles qu'au bout de plusieurs générations. Au bout d'un certain nombre de générations, une fois qu'une solution optimale est obtenue, le processus peur s'arrêter.

Les algorithmes génétiques sont simples sur le plan opérationnel et représentent un bon choix pour résoudre des problèmes avec un grand espace de recherche, dont on connait peu les caractéristiques. En tant que tels, ils ont été appliqués efficacement à de nombreux problèmes du domaine de l'optimisation (tels que la résolution de problèmes NP-complets [12]), de planification, d'ordonnancement, de conception, etc.

3.7.1.1 Principe

Le processus qui en découle est imité de la biologie. Le processus de sélection naturelle commence par la sélection des meilleurs individus au sein de la population. Ils produisent une progéniture qui hérite des caractéristiques des parents et sera ajoutée à la prochaine génération. Si les parents sont plus adaptés au besoins de l'environnement, leur descendance sera meilleure que les parents et aura de meilleures chances de survie, et à la fin, les individus les plus aptes seront trouvés. Cette notion peut s'appliquer à un problème de recherche. Nous considérons un ensemble de solutions à un problème et sélectionnons l'ensemble des meilleures d'entre elles. Un algorithme génétique se compose généralement de cinq phases :

— La création d'une population de base générée aléatoirement. La popu-

lation initiale est constituée d'un ensemble d'individus qui sont des solutions potentielles générés de manière aléatoire ou heuristique. Dans un algorithme génétique classique, chaque individu est représenté par une chaîne binaire de longueur fixe de bits (un chromosome) qui encode les paramètres du problème. Cette chaîne peut être décodée pour donner les valeurs entières de ces paramètres.

— **La sélection** La sélection détermine quels sont, au sein de la population, les individus les plus adaptés, ayant les meilleurs résultats.

— **Le croisement** Le croisement imite la reproduction. Deux chromosomes "parents" se croisent et échangent un partie de leurs chaines, donnant naissance à de nouveaux chromosomes.

— **La mutation** Un gène peut muter de manière aléatoire au sein d'un chromosome. La probabilité est souvent faible. Cela sert à ne pas tomber dans un optimum local, ou une convergence prématurée.

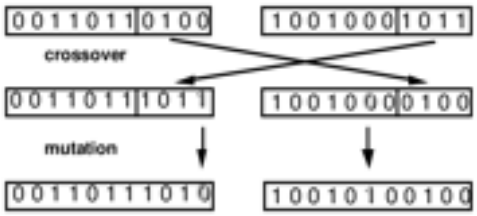

Figure 3.28 – Croisement et mutation

— **L'évaluation** Les algorithmes génétiques requièrent que les individus de la population puissent être différenciés en fonction de leur qualité. Les membres les plus aptes ont une probabilité plus élevée de participer aux phases de sélection et de reproduction. La fonction d'évaluation détermine à quel point un individu est capable de rivaliser avec d'autres individus. La fonction d'évaluation est mesurée en décodant un chromosome et en utilisant les paramètres décodés comme entrée de la fonction objectif. La valeur renvoyée par la fonction objectif est utilisée comme valeur de fitness.

L'algorithme se termine (converge) si, dans la population des individus est suffisamment proche de la solution recherchée, d'après l'évaluation par la fonction de fitness. L'algorithme génétique peut aussi fournir un ensemble de solutions.

Soit P(t) la population, l'algorithme est le suivant :

Listing 3.6 – Le processus itératif d'un algorithme génétique

```
DEBUT
Generer la population initiale
Calculer le fitness
REPETER
     Selection
     Crossover
     Mutation
     Calcul de la fitness
TANT QUE la population ne converge pas
ARRET
```

3.7.1.2 Codage

La première et la plus importante étape de la conception de l'algorithme génétique est le choix de la représentation du problème à résoudre. L'encodage des individus est une partie cruciale d'un algorithme génétique car de cette représentation dépendent d'autres caractéristiques de l'algorithme : dont la taille de l'espace de recherche, l'efficacité des opérateurs génétiques, etc... Une représentation inadaptée peut donner des performances médiocres.

Cette représentation peut encoder l'apparence, le comportement, l'action, les qualités des individus. Il existe plusieurs types de représentations :

Représentation binaire : la plus simple et la plus courante, le génotype se compose simplement de chaînes de bits. Pour certains problèmes lorsque l'espace de solution se compose de variables de décision booléennes, la représentation binaire est naturelle.

Représentation réelle pour les problèmes où nous voulons définir les gènes en utilisant des variables continues plutôt que discrètes, la représentation à valeur réelle est la plus naturelle.

Représentation entière : pour les gènes à valeur discrète, nous ne pouvons pas toujours limiter l'espace de solution aux valeurs binaires. Différents

choix peuvent être encodés par une suite d'entiers par exemple.

3.7.2 L'algorithme génétique appliqué à la génération automatique de règles

3.7.2.1 Population initiale

Une population initiale est généralement créée avec valeurs aléatoires pour tous les gènes. Une petite population peut ne pas être représentative de l'ensemble de l'espace de recherche mais l'effort de calcul pour chaque génération est plus faible. Une grande population implique un coût de calcul plus élevé pour chaque génération, mais peut converger en utilisant moins de générations. L'alternative que nous choisissons est d'utiliser une petite population et une probabilité de mutation relativement élevée au début pour s'assurer qu'une plus grande partie de l'espace de recherche du problème est explorée.

La population initiale ici est définie de manière aléatoire et est composée de 30 chromosomes. Chaque chromosome contient des gènes qui seront utilisés pour créer les formes architecturales. Tous les individus avec leurs gènes sont évalués en utilisant la validation par rapport à des contraintes globales et spécifiques.

3.7.2.2 Selection de la population

Le processus de choix des individus qui passeront à la génération suivante et se reproduiront s'appelle la sélection. Grâce à la capacité de sélection, il est possible de conserver les meilleurs chromosomes pour les générations futures afin de découvrir une solution optimale. Il empêche également d'éventuels bons chromosomes de se perdre dans la population, et choisit les individus qui se reproduiront par croisement et mutation.

Une approche possible qui consiste à sélectionner les chromosomes N qui survivront et feront partie de la nouvelle population évolutive peut être décidée par classement, où les N individus les plus adaptés de l'ensemble des parents et de la progéniture sont choisis.

L'élitisme consiste en la sélection d'un individu ou d'un groupe d'individus qui survivront pour la prochaine génération sans subir de croisement et de mutation. Dans ce cas, seul le meilleur chromosome de la population est sélectionné. C'est un mécanisme qui copie la moitié des parents les plus adaptés et la moitié de la progéniture la plus adaptée vers la nouvelle population évolutive. Une méthode hybride (élitisme et classement) déplace les 10% de parents les plus adaptés directement dans la nouvelle population. Les individus restants sont choisis parmi des individus plus ajustés entre les parents restants et la progéniture.

La sélection proportionnelle à la fitness, se fait en fonction de la forme physique de chaque individu. La distribution proportionnelle probabiliste est calculée dans l'équation suivante où P(i) est la probabilité que l'individu i soit sélectionné et N est la taille de la population.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{3.1}$$

Avec une sélection proportionnée à la fitness, il y a une chance que certaines solutions moins adaptées survivent au processus de sélection. Même si la probabilité de survie des solutions les plus faibles est faible, elle n'est pas nulle, ce qui signifie qu'il est toujours possible qu'elles survivent ; ceci est un avantage, car il y a une chance que même les solutions moins adaptées puissent avoir certaines caractéristiques qui pourraient s'avérer utiles après le processus de recombinaison.

Ces sélections sont adaptées pour un simple objectif mais nous introduisions de multiples objectifs dans notre système. De nombreux algorithmes évolutifs multi-objectifs ont été créés ces dernières années. Le NSGA-II est composé de deux parties principales : une partie solution de tri rapide non dominé et la préservation de la diversité de la solution, avec une complexité de calcul de $O(MN^2)$ où M est le nombre d'objectifs et N est la taille de la population. L'approche de sélection est élitiste et crée une nouvelle génération en combinant les populations de parents et de descendants et en sélectionnant les meilleures solutions (en ce qui concerne la fonction de fitness et la bonne propagation des solutions dans l'ensemble des solutions obtenues).

3.7.2.3 Mutation

La mutation est très importante pour introduire un nouveau matériel génétique de manière aléatoire. Il permet de rechercher un large éventail de solutions afin de trouver la solution optimale. Ici, la probabilité de mutation a une valeur élevée au début, puis elle diminue à mesure que le système évolue. De cette façon, nous élargissons nos recherches au début et plus tard, lorsque les chromosomes commencent à converger vers la solution optimale, nous évitons de faire de grands changements.

3.7.2.4 Croisement

Le croisement est un opérateur génétique utilisé pour combiner les informations génétiques de deux parents pour générer une nouvelle progéniture. Afin de faire le croisement, de nouveaux descendants sont générés en utilisant les chromosomes sélectionnés par le processus de sélection multi-objectif. Il existe de nombreuses méthodes pour réaliser le croisement. Ici, nous choisissons le croisement en un point : à partir d'un index qui est défini de manière aléatoire, nous découpons deux chromosomes parents en deux, nous commutons ces parties entre les deux chromosomes "parents" sélectionnés en créant deux nouveaux chromosomes descendants.

3.7.2.5 Critères d'arrêt

L'algorithme génétique doit être capable d'arrêter l'évolution lorsque l'objectif est atteint. A cet effet, nous avons les critères de convergence suivants :

1. Celui que l'on utilise le plus souvent dans nos travaux : l'itération s'arrête lorsque la variation de la fitness minimale ou de la moyenne au cours de plusieurs générations reste en dessous d'un seuil donné.
2. Si cela ne se produit pas, l'itération s'arrête lorsqu'un nombre maximal de générations précédemment établi est atteint : défini à 1000 générations dans notre cas.

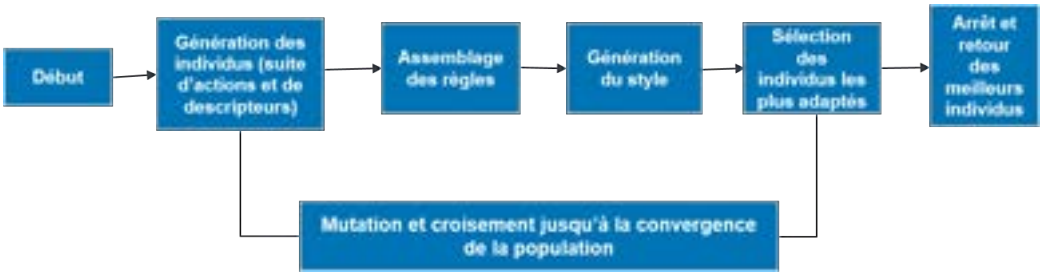

Figure 3.29 – Schéma de l'algorithme génétique

3.7.2.6 Codage des individus

Après avoir considéré les règles que nous avons déjà, nous avons opté pour l'introduction d'actions intermédiaires, spécifiquement pour l'encodage de la structure d'un individu type, pour faire le lien avec les règles grammaticales. Dans la modélisation d'une forme, deux principes sont repris : une action faite sur la forme et une description de la disposition des formes enfants ou arrangement, appliquées avec des paramètres numériques. Le but étant d'obtenir les combinaisons les plus optimales correspondant à un ou plusieurs combinaisons d'objectifs stylistiques recherchés. Comme la procédure de génération consiste à effectuer une succession d'actions, les individus sont encodés comme une succession d'actions, arrangements et paramètres numériques.

Cela rend la structure d'une chaine d'actions bien plus lisible, et moins aléatoire. Cette chaine crée une séquence de taille fixe qui forme le génotype. Chaque action est associée à un nombre, de même que chaque arrangement, et enfin un paramètre pour cet arrangement. La séquence d'actions et l'ordre d'exécution évolue à chaque génération. Chaque individu est donc une suite de nombres représentant trois types d'informations :

1. Une action
2. Un arrangement spatial
3. Des paramètres numériques

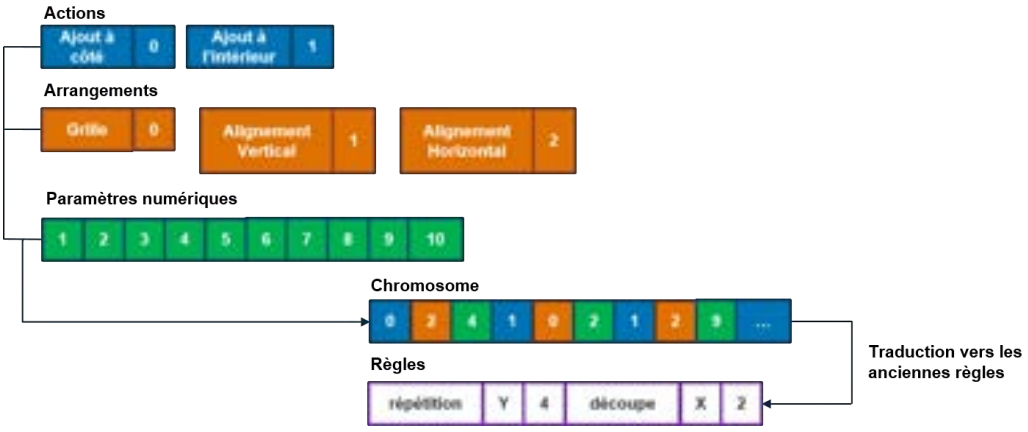

Figure 3.30 – La création des règles

3.7.2.7 Actions intermédiaires

Comme mentionné plus tôt, des actions intermédiaires sont introduites, faisant le relai avec nos règles grammaticales. Notre but est l'obtention d'une chaine d'actions décrivant les actions appliquées à une forme jusqu'à l'obtention d'un résultat final répondant aux critères, tout en gardant à l'esprit que ce résultat doit être adapté à l'architecture.

Les arrangements introduits sont : la répétition verticale, la répétition horizontale, et la grille. Ces arrangements sont introduits avec les actions : ajoute et contient, qui sont les deux seules actions intermédiaires. La fonction ajout réécrit les règles précédentes appliquées sur la dernière forme concernée. La fonction contient agit comme les règles précédentes et ajoute simplement une autre règle. Les paramètres numériques régissent le nombre de répétitions et de divisions. Toutes les formes sont des marqueurs à ce stade. Les symboles terminaux sont introduits à la fin de la génération.

3.7.2.8 Evaluation

L'algorithme génétique doit faire évoluer les chromosomes en une disposition réalisable, c'est-à-dire conforme à toutes les règles de conception et sans chevauchement ou incohérence entre le positionnement des pièces, ce qui sera considéré comme une solution avec une fitness optimal. Ainsi, le non-respect

d'une règle provoque une pénalisation vers la valeur maximale.

Pour définir une fonction de fitness quantifiable, différentes pénalisations ont été mises en oeuvre sous forme de quantités positives afin de maximiser la fitness, dont la valeur optimale tendrait vers 0. Pour cela, les règles présentées étaient mis en oeuvre en les transformant en des décisions et des points de pénalité ont été ajoutés à la pénalisation totale du style.

Une solution optimale aura une valeur minimale, ce qui signifie que toutes les règles de mise en forme sont respectées et qu'une forme désirable a été obtenue. Toute infraction à une règle entraîne une pénalité de 100.

A chaque objectif est associé une fonction qui évalue la proximité de l'apparence obtenue avec cet objectif stylistique. Dans son ensemble, ces fonctions mesurent la similarité avec un motif précis. Notre but étant de trouver l'ordre d'éxecution permettant d'optimiser chaque objectif sans qu'il y ait une dominance d'un objectif particulier. La contrainte de multiples objectifs nous impose le calcul d'un ensemble Pareto-Optimal, qui implémente le principe de non-dominance des solutions et fournit une solution optimale de telle sorte que certains objectifs ne soient pas privilégiés au dépend d'autres objectifs. La sélection des individus les plus adaptés est effectuée avec l'algorithme NSGA-II. Deux éléments sont analysés au cours de l'évaluation du style obtenu, une matrice listant les terminaux dans le même ordre où ils sont disposés sur la façade test, et l'arbre de formes.

Deux types d'objectifs sont évalués : les objectifs globaux et les objectifs esthétiques. Les objectifs globaux sont mis en place pour vérifier la cohérence des formes obtenues d'un point de vue architectural, les mesures obtenues sont de ce fait ajoutées à tous les objectif esthétiques. Les objectifs globaux incluent : la vérification des dimensions de manière à ce que les fenêtres ne soient pas trop petites ou grandes, un nombre minimal d'éléments d'une façade, la mise en place d'une préférence telle que le nombre de subdivisions d'une itération soit inférieure à celui de l'itération précédente, ce qui peut engendrer un nombre trop conséquent de subdivisions totales au final.

Il y a six objectifs esthétiques :

Symétrie La symétrie est mesurée directement sur la matrice des terminaux : la symétrie verticale en comparant chaque élément en partant des

extrémités des lignes de la matrice vers le centre, et la symétrie horizontale en utilisant le même concept appliqué aux colonnes de la matrice des terminaux. Chaque inégalité incrémente la fonction d'évaluation.

Figure 3.31 – Évaluation de la symétrie horizontale

Uniformité L'évaluation de l'uniformité mesure la 'distance' entre la matrice des terminaux $A = (a_{i_j})$ et de matrices $B = (b_{i_j})$ composées chacune d'une différente ligne de cette matrice des terminaux, répétée le long des colonnes.

$$d(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i_j} - b_{i_j}|$$

La plus petite mesure de distance est considérée car la matrice des terminaux est alors la plus proche de cette matrice, donc les lignes sont uniformes entre elles.

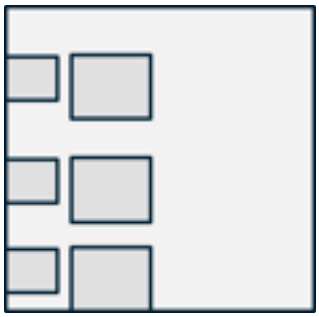

Figure 3.32 – Uniformité

Réctilinéaire Le ratio entre la largeur et la hauteur doit être inférieure à un seuil défini pour chaque terminal.

Figure 3.33 – Réctilinéarité

Monolithisme Le monolitisme mesure combien de terminaux sont obtenus au final, un plus grand nombre suggérant un important fractionnement, et l'absence d'un monolithe.

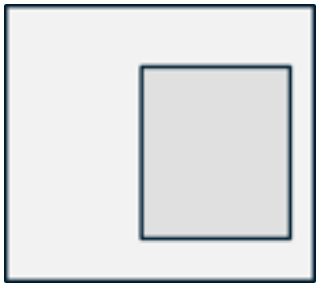

Figure 3.34 – Monolithisme

Fragmentation La fragmentation va chercher à maximiser le nombre de terminaux au final dans le respect des seuils des mesures des objectifs globaux.

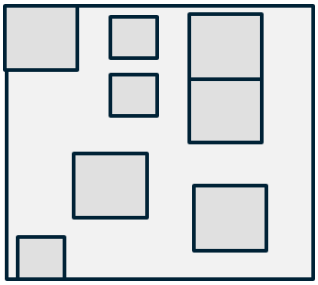

Figure 3.35 – Fragmentation

3.7.2.9 Processus

De la même manière que dans [23] le même processus de changement de style via la transformation de la grammaire est repris mais nous introduisons l'utilisation d'algorithmes génétiques. Dans un premier temps, une population d'individus encodés en enchainement d'actions, de descripteurs et de paramètres est donc obtenue, cet enchainement est traduit en ensemble de règles normales et une grammaire est assemblée avec la structure décrite ci-dessus.

Selon les objectifs définis, une évaluation est faite pour chaque objectif sur les individus. De nouvelles grammaires sont générées à partir des grammaires précédentes via les règles de mutation et de croisement.

Les générations sont faites sur un simple objet de type façade, et l'analyse de l'évaluation effectuée sur l'arbre résultant, qui contient assez d'informations pour les calculs qui s'en suivent. Le module de création de règles opère de la manière suivante : les règles sont interprétées en faisant du LHS un marqueur temporaire du type 'shape1'. Chaque marqueur se termine par un nombre croissant à chaque itération, indiquant le nombre d'itérations effectuées jusque là. A la fin de l'écriture de l'ensemble de règles, de nombreux RHS seront des non-terminaux, ils sont remplacés par des terminaux en utilisant un choix pondéré suivant la logique suivante :

— Au début, on a une façade
— L'introduction d'un marqueur pour un emplacement d'ouverture sera plus commune au début de l'itération.
— S'en suit les marqueurs pour une fenêtre
— Et enfin l'introduction d'autres murs adjacents ou à l'intérieur dans l'optique de compléxifier la forme de la fenêtre (remplacer une baie

vitré par 2 plus petites fenêtres)
Le choix d'instancier un terminal donné est régi par cette probabilité. La probabilité qu'un mur apparaisse par exemple est plus probable au début, et inversement proportionnelle au niveau d'itération (extrait du nom de la forme). Cette tendance s'inverse vers la fin de l'itération, une occurence des murs est plus commune, ...

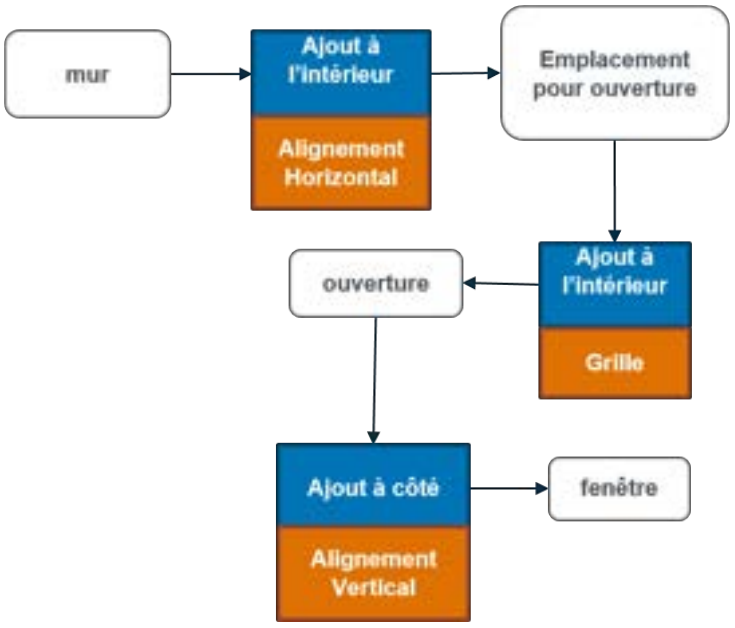

Figure 3.36 – Processus de création des règles

3.7.3 Expérimentation et résultats

Cette section présente des expériences menées pour évaluer les capacités de l'algorithme génétique à résoudre le problème de création de style. L'algorithme génétique utilisé dans nos expériences opère sur une population de taille 50. La probabilité de croisement de 0.7 été choisie et la mutation est de 0.5 à 0.3. Nous trouvons que la population converge autour de 30 itérations.

Les résultats suivants ont été produits en utilisant une implémentation Python codant directement l'algorithme génétique précédemment défini. Le

système a été testé sur les différentes combinaisons d'opérateurs.

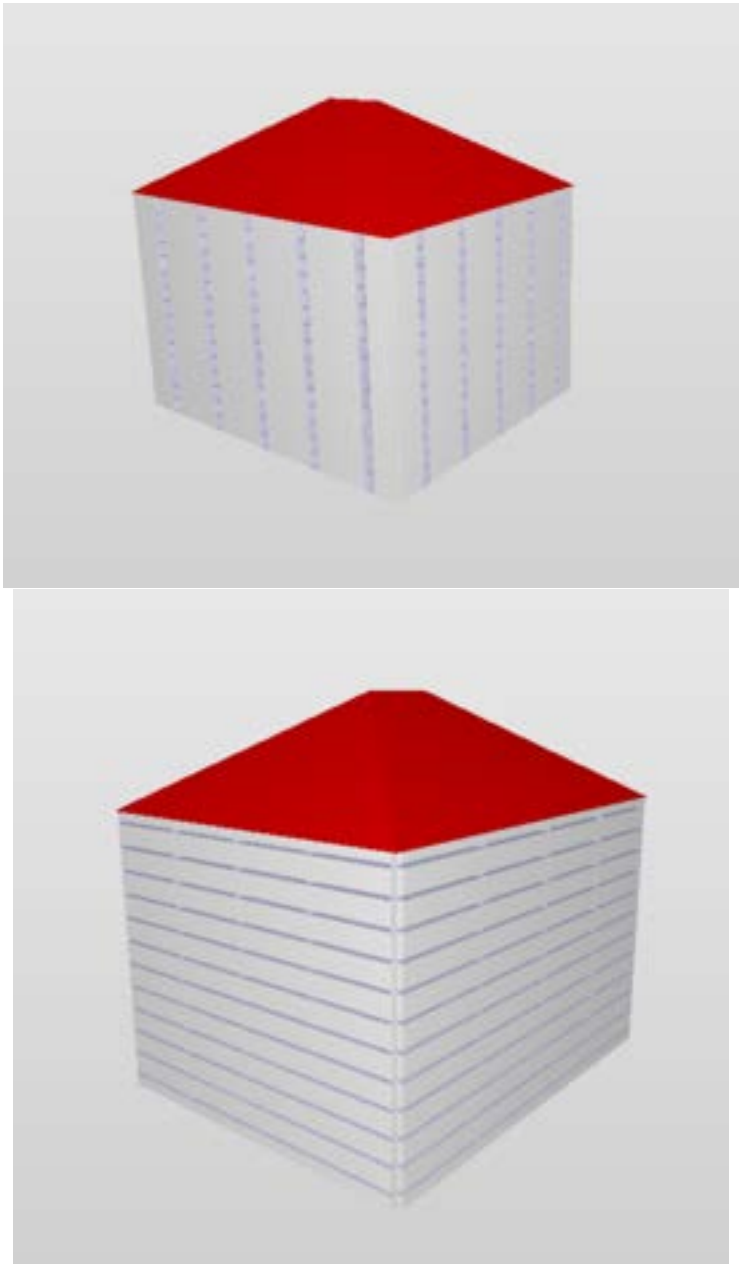

Figure 3.37 – Bâtiments obtenus pour le style "symétrie horizontale"

Figure 3.38 – Chromosome pour le style "symétrie horizontale"

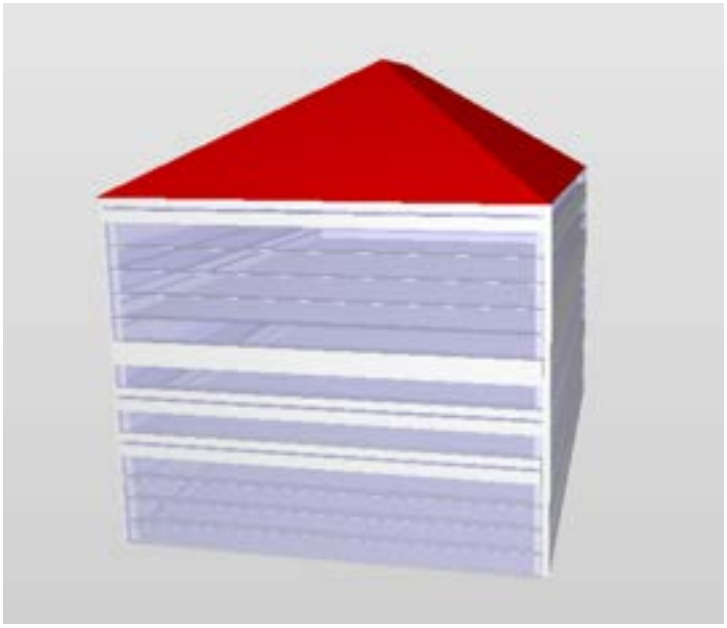

Figure 3.39 – Bâtiment obtenu pour le style "symétrie verticale"

Figure 3.40 – Chromosome pour le style "symétrie verticale"

Figure 3.41 – Bâtiment obtenus pour les styles "symétrie horizontale et verticale"

Figure 3.42 – Chromosome pour le style "symétrie horizontale et verticale""

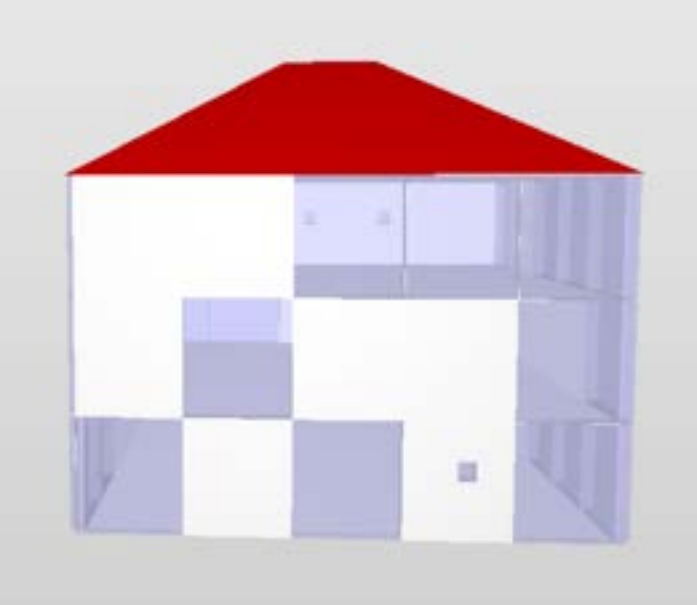

Figure 3.43 – Bâtiment obtenus pour les styles "rectiligne et monolithique"

Figure 3.44 – Chromosome pour le style "rectiligne et monolithique"

L'utilisateur commence par fournir un objectif ou une liste d'objectifs désirés. Durant l'éxecution de l'algorithme, à chaque itération est créée une série d'instructions et un ensemble de règles est généré à partir de ces instructions. Cet ensemble de règles génère une façade test (via l'interpreteur de grammaire) qui est évaluée selon les critères esthétiques fournis en entrée. Les instructions sont ajustées avec l'algorithme génétique jusqu'à la convergence vers un résultat désirable. L'évolution s'arrête lorsqu'une façade optimale est trouvée ou l'algorithme converge vers un optimal.

A la fin, le meilleur candidat est sélectionné, et l'ensemble de règles résultant génère le bâtiment correspondant.

Chapitre 4

Discussion

Sommaire

4.1 Résumé

Notre approche pour aborder la génération et la complexité des modèles de villes virtuels 3D consiste à concevoir des méthodes de génération automatique de bâtiments, abordées à partir de deux fronts différents mais tous deux basés sur la production de grammaires. Pour cela, nous avons appliqué et combiné des principes généraux issus d'algorithmes génétiques et grammaires de formes pour générer des jeux de règles de manière automatique. La raison principale en était de trouver un juste milieu entre autonomie totale et génération assistée. Les résultats indiquent que notre système a pu créer un certain nombre de résultats satisfaisants sur différents fronts.

Les données de construction peuvent être fournies de plusieurs manières : des fichiers shapefiles, en remplissant progressivement une base de données

de connaissances, en combinant les deux ou avec un algorithme génétique. La suite comprenait trois étapes majeures :

1. La création des règles
2. L'interprétation de la grammaire et autres modifications qui peuvent se faire sur l'arbre de forme.
3. L'exportation vers le modèle IFC qui se fait en créant le lien entre l'arbre de forme et les plusieurs instances offertes par le modèle IFC. Le résultat est assez lourd, vu la complexité du modèle, mais également complexe.

La création des règles en particulier peut se faire de deux manières que l'on va expliquer dans les paragraphes suivants.

4.1.1 Base de connaissance

La base de connaissance permet un certain nombre de manipulations et donne le contrôle au niveau de la division des formes individuelles via une interface. La création des règles, et de même l'alimentation de la base de connaissance elle-même peut se faire de manière intuitive à l'aide de la conception de dessins et de simples choix multiples, offrant un contrôle plus aisé sur le processus.

4.1.2 Algorithme génétique

L'algorithme génétique applique le principe de la transformation grammaticale, transformant celle-ci jusqu'à obtenir un résultat satisfaisant.

Le problème qu'il devait résoudre consistait à trouver les meilleurs compromis entre tous les objectifs esthétiques. L'objectif étant de trouver un bon front de Pareto pour trouver les solutions qui, tout en ayant une bonne performance par rapport à un objectif, aient aussi la meilleure performance possible par rapport à un autre, compte tenu de la priorité donnée au premier, et vice-versa. Cette technique est flexible et permet la définition formelle de plusieurs styles dans le langage de conception d'une grammaire.

Les itérations sur l'algorithme ont été réalisés avec six fonctions d'objectif, dont les valeurs sont à minimiser pour tous les objectifs. La progression de

la fonction d'évaluation suivant la recherche dans deux exemples est montrée dans les figures suivantes. Parce que le problème était assez complexe, en raison du grand nombre de variables dans le problème et de l'utilisation de plusieurs fonctions d'objectifs, la taille de la population était de 50 individus. Les résultats convergent en environ 30 itérations.

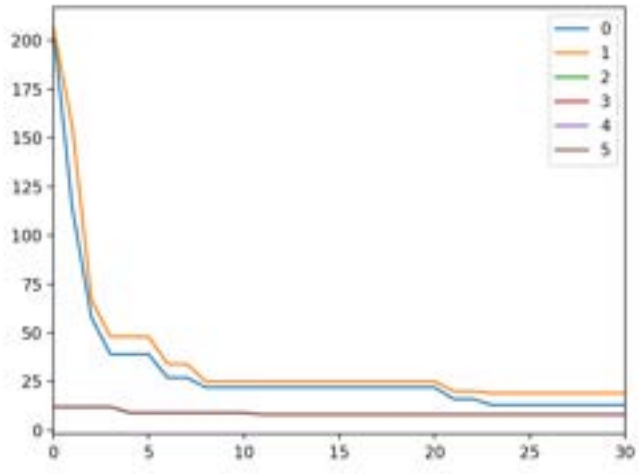

Figure 4.1 – Convergence avec les styles "symétrie verticale" et "symétrie horizontale"

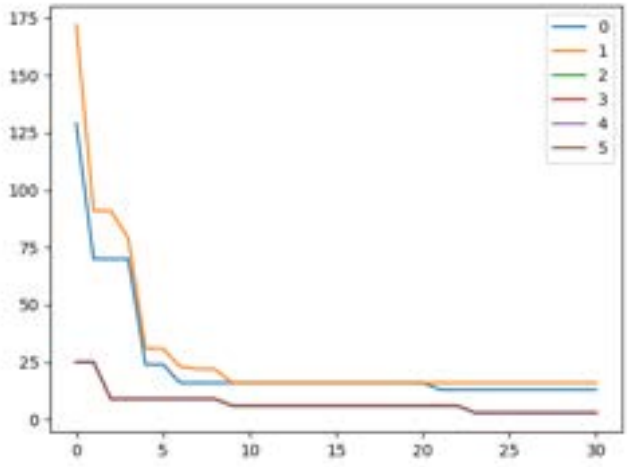

Figure 4.2 – Convergence avec les styles "symétrie verticale" et "rectilinéarité"

4.2 Évaluation qualitative

Dans cette section, nous allons répondre aux questions abordés dans la thèse.

1 - Création d'une méthode de modélisation procédurale plus orienté dans la génération d'un type de contenu généralisé

Le système est orienté vers une utilisation plus générale. Le système de conception générative, associé aux méthodes de création automatiques de règles, a pu créer une variété de formes architecturales qui répondent aux différents objectifs de conception en termes d'esthétisme.

Cette génération peut être considérée comme un mécanisme dont le but n'est pas seulement d'atteindre un objectif spécifique, mais aussi de proposer des configurations de bâtiment et de fonctionner comme une aide à la conception. Les formes générées dans ces expériences sont le résultat des données initiales, des règles et des contraintes appliquées. Des conditions initiales différentes conduiraient à l'émergence d'autres solutions, ce système peut

ainsi être un outil puissant pour les concepteurs pour étudier rapidement des conceptions alternatives ou pour générer des villes que les concepteurs pourraient affiner plus tard, l'option permettant à l'utilisateur de modifier des terminaux spécifiques peut améliorer considérablement le niveau de contrôle des concepteurs.

2 - Caractère intuitif et contrôle utilisateur
Bien que les règles puissent être créées et éditées manuellement, notre principal objectif est de fournir à l'utilisateur un moyen simple et rapide pour créer et modifier des prototypes, ainsi qu'une méthode prévisible de gérer les modifications. Cet outil permet de modéliser un vaste panel de caractéristiques. Un utilisateur sans aucune expertise technique serait en mesure d'utiliser correctement l'application, car les détails de mise en œuvre ne sont pas pertinents pour l'utilisateur. De plus, les étapes nécessaires à la génération d'un bâtiment suivent un assistant de création étape par étape qui est plus familière à la plupart des utilisateurs.

3 - Automatisation la procédure de modélisation basée sur les grammaires en trouvant un juste milieu.

Le degré de contrôle offert à l'utilisateur se situe au niveau de la fourniture de paramètres statiques à chaque module, l'utilisateur peut remplacer des terminaux individuels et régénérer le contenu de chaque module. Le choix de conception consiste soit à créer une architecture avec un algorithme génétique dynamique, moins précis et bien plus automatisé et rapide soit à utiliser un générateur de ville capable de produire des instances en alimentant une base de données, avec la possibilité de concevoir manuellement des formes et autres paramètres, un processus plus long mais dans ce cas les résultats sont extrêmement prévisibles. Le résultat final peut être modifié en dans les deux cas.

4 - Lien entre SIG et BIM

Nous avons exploré la possibilité de générer des modèles IFC avec la modélisation procédurale. Les informations SIG peuvent être stockées dans la base de données et transférées au BIM. Ce système est un outil intuitif pour intégrer l'utilisation du BIM et la gestion de l'information peut servir de base à un BIM enrichi et faciliter la gestion de projets de construction.

5 - Faire coopérer différentes méthodes de création dans la génération d'un modèle complexe.

Le système fait entrer en jeu des principes allant des grammaires aux bases de données, du dessin manuel aux algorithmes génétiques, dans le but d'obtenir un modèle IFC. Les résultats se sont avérés en général utiles pour comprendre comment les compromis entre des objectifs contradictoires influencent les solutions obtenues.

4.3 Comparaisons

Notre première approche peut être liée à deux méthodes mentionnées dans des travaux connexes, et fait plus ou moins le lien entre Biljecki, Ledoux et Stoter [5] et Müller et al. [32] Nous gardons l'expressivité des jeux de règles générées automatiquement et stockons les informations dans un fichier de jeux de données avec une représentation multi-LOD possible.

Nous utilisons toutefois une approche différente et différents ensembles de données. Notre moteur génère des données de connaissances géographiques à partir d'une interface utilisateur, avec un jeu de règles correspondant, et un autre module génère le modèle correspondant. Nous avons choisi d'utiliser la norme IFC, un modèle avec une cohérence spatio-sémantique complète. Bien que le fichier de jeu de données puisse être utilisé pour des modèles purement fictifs, il peut être directement créé à partir des données OSM.

L'interface graphique rend la génération de règles plus rapide et plus facile pour les utilisateurs non-experts, les règles et les fichiers xml sont extensibles, ainsi que le modèle généré. L'utilisateur a le contrôle sur le processus de transformation grammaticale.

L'extérieur et l'intérieur complet peuvent être générés, y compris les pièces et les ouvertures qui sont construites donnant un niveau supplémentaire de détails granulaires, par opposition à la première approche dans [5], les règles que nous utilisons sont plus simples par rapport à celles de CityEngine [32], mais par conséquent, nous n'avons pas les formes résultantes plus complexes, en particulier les formes courbes, mais le compromis de l'interface offerte est

particulièrement important pour la génération rapide de bâtiments personnalisés.

D'un autre côté, sur l'introduction des algorithmes génétiques, bien que déjà utilisé pour les transformations grammaticales dans d'autres domaines [23], ou dans la création d'architecture [8], la combinaison des deux pour l'optimisation d'objectifs esthétiques sur les façades est relativement nouvelle pour l'architecture.

Précédemment, *EiForm* [37] a combiné la grammaire de formes et l'algorithme génétique pour créer des structures en treillis en trois dimensions, projetées sur une surface fournie par l'utilisateur. Ce système produit spécifiquement des modèles de treillis triangulés, en évaluant la masse structurelle et l'intention de conception formulée avec un modèle géométrique.

Shape Evolution [11] combine également les deux, en prenant un ensemble de critères de conception en entrée, et produit des combinaisons de blocs 3D formant un bâtiment, *Shape Evolution* se concentre sur le modèle de masse et en optimise des aspects tels que la hauteur, l'empreinte au sol, ...

Contrairement à Caldas et Norford [8], nous avons mis l'accent sur les formes et esthétiques, suivant l'intention de Khan et Chase [23] dans l'utilisation de transformations grammaticales pour diriger le changement de style. Suivant ce principe, nous avons développé des mesures pour représenter les qualités esthétiques des éléments de grammaire. Ces mesures ont permis de comparer les descripteurs adjectivaux des grammaires avec les styles et designs.

4.4 Limites

L'un des défis lors de l'étude de cas était la préparation de l'ensemble de données de construction basé sur le SIG de la ville avec les informations requises sur les caractéristiques du bâtiment réels. L'agrégation de plusieurs types de données dans un ensemble de données de construction de ville nécessite un travail considérable. La qualité des données est devenue un problème important lors de l'exécution de la simulation. Les informations sur la hauteur du bâtiment et le nombre d'étages ont été réconciliés en les comparant

avec celles de Google 3D Map et des informations en ligne.

Du coté de l'algorithme génétique, il est impossible de prédire de manière exacte l'apparence qu'une transformation grammaticale donnera : il serait intéressant de renforcer les descripteurs adjectivaux en les contraignant un peu plus dans les grammaires. Un tel contrôle sur le processus de transformation grammaticale est encore souhaitable.

Enfin, tout au long du développement, nous trouvions souvent que le générateur produisait un contenu visuellement suffisant, mais dont le processus de génération était lent, et cela est du aux opérations géométriques rendant le modèle en lui-même lourd. La performance qui était nécessaire pour exécuter la génération de la ville est à améliorer.

Chapitre 5

Conclusion et perspectives

Lorsque les bâtiments partagent un style architectural, leur similitude réside dans la recherche de leurs caractéristiques de base sous-jacentes. Les grammaires de forme y parviennent en clarifiant la structure commune et en dépeignant toutes les divisions hiérarchiques que cette structure peut traverser. Dans le processus de construction de la ville en particulier, un problème se pose dans le stockage d'énormes ensembles de données.

Les méthodes procédurales fournissent des moyens puissants pour générer des structures géométriques complexes mais sont difficiles à contrôler. Compte tenu de la spécification de haut niveau de la production souhaitée, l'algorithme crée une grammaire conforme à la spécification. Nous avons présenté une solution pour contrôler la création de ces modèles procéduraux basés principalement sur le contrôle des grammaires.

Comme un seul jeu de règles peut générer de multiples variations des styles urbains, avec des détails plus complexes, nous offrons une alternative avec deux différentes solutions pour aborder le problème, et intuitivement créer ce jeu de règles : alimenter une base de connaissance, ou automatiquement créer un style.

Nos travaux visaient à explorer l'utilisation des algorithmes procéduraux pour générer de manière semi-automatique des villes 3D modernes en se basant sur des travaux antérieurs et des techniques procédurales existantes dans plusieurs domaines afin de concevoir et de proposer un système complet

capable de générer de telles solutions. Le système que nous avons proposé a été implémenté et démontré sous la forme d'une application qui combine plusieurs techniques telles que les grammaires de formes, les algorithmes génétiques et les niveaux de détails pour obtenir une génération urbaine satisfaisante pour l'utilisateur.

Les contributions de la thèse s'articulent autour de deux axes que nous avons présenté dans les précédents chapitres : le lien entre l'architecture générée par ordinateur, le BIM et le SIG via une interface de création, et la proposition d'un ensemble de méthodes de personnalisation et de génération de règles grammaticales afin d'apporter un résultat visuel plus pertinent.

À travers ces recherches, nous avons apporté des réponses aux problématiques de la thèse.

Nous avons créé une structure de données pour unifier les données SIG avec les informations d'apparence fournies par l'utilisateur et au final offrons la possibilité de convertir les données créées en un format standard : IFC.

Après avoir implémenté une grammaire de formes pour mieux comprendre et maîtriser le fonctionnement de cette dernière, nous nous sommes penchés les méthodes de génération d'une grammaire.

Au niveau de la génération, deux implémentations ont été créées : la première s'appuyant sur des données statiques, et des petites ou moyennes variations dans un cadre précis et contrôlé, offrant ainsi un résultat prévisible. La seconde, basée sur les algorithmes génétiques troque une partie de ce contrôle contre l'émergence de nouveaux designs.

Bien que loin d'être d'une qualité de production commerciale, le système fournit un aperçu utile de diverses techniques applicables à la génération urbaine et comment elles peuvent être combinées. La conception de bâtiments avec ce système s'avère être efficace dans un cas ou dans l'autre, fournissant un langage puissant dans lequel exprimer les formes architecturales.

La système est assez générique et simple pour une compréhension facile, mais peut être utilisé pour formuler des règles significatives pour briser des formes simples en plus complexes, et faire évoluer la conception de la forme

dans des styles de construction plus spécifiques, en ajoutant éventuellement de plus en plus de détails sur le modèle de masse du bâtiment.

Modifier quelques paramètres ou ajouter quelques détails peut déplacer toute la scène d'un style particulier à l'autre, et le système donne un bon compromis entre la simplicité et la qualité visuelle.

Cependant, le logiciel doit probablement mûrir un peu plus, les principales limitations étant la qualité des données en entrée, le besoin d'un contrôle plus poussé sur l'algorithme génétique, et la lourdeur de l'export final, imposé par le format d'export IFC. Dans son état actuel, on démontre la capacité de générer le contenu inclus dans la portée de notre étude, il serait intéressant de mettre l'accent sur le système, en incluant plus d'aspects à générer, tels que les infrastructures de transport public.

Chapitre 6

Annexe

Exemple de fichier XML généré avec la base de connaissance

```
<earth>
<info color="1;1;1" material="BricksBasic" nbfloor="1" roofangle
    ="15" rooftype="flat" />
<country designation="France">
<info color="pink" dist="5" lat="47.3833300" lon="0.6833300"
    material="BricksBasic" nbfloor="1" roofangle="15" rooftype="
    gabbled" />
        <region designation="Midi-Pyrenees">
                <info color="grey" dist="1.1" lat="43.600000"
                    lon="1.433333" material="BricksBasic" nbfloor
                    ="1" roofangle="15" rooftype="flat" />
                <town designation="Toulouse">
                        <info color="yellow" dist="0.8" lat
                            ="43.600000" lon="1.433333" material
                            ="BricksBasic" nbfloor="3" roofangle
                            ="15" rooftype="flat" />
                        <district designation="UPS">
                                <info color="blue" dist="0.03"
                                    lat="43.560397" lon
                                    ="1.468820" material="
                                    OfficeGlasses" nbfloor="4"
                                    roofangle="0" rooftype="flat"
                                     />
                        </district>
                        <district designation="Centre-Ville">
                                <info color="white" dist="0.03"
                                    lat="43.603236" lon
```

```
                                    ="1.444659" material="
                                    BricksBasic" nbfloor="3"
                                    roofangle="15" rooftype="
                                    hipped" />
                            </district>
                        <appearance><values bdRooftype="mansard" color="
                            black" doorCol="brown" floorsize="4"
                            maxFloors="1" maxHeight="5" maxrotation=""
                            minFloors="1" minHeight="5" openingCol="black
                            " proba="" roofColor="black" windowfcolor="
                            black" windowheight="tall" windowmaterial="
                            glass" windowstyle="simple" windowwidth="2"
                            /><opening><split axis="Z" children="
                            nt4547681976|nt4629111416" parent="
                            nt4547681752" splitvalues
                            ="0.34523809523809523,0.6547619047619048" /><
                            elements parent="nt4547681752" terms="
                            nt4547681976,nt4629111416," /></opening><
                            ruleset file="data.xml" /></appearance><
                            appearance><values bdRooftype="hip" color="
                            floralwhite" doorCol="brown" floorsize="4"
                            maxFloors="1" maxHeight="23" maxrotation=""
                            minFloors="1" minHeight="5" openingCol="white
                            " proba="" roofColor="lightslategray"
                            windowfcolor="seashell" windowheight="short"
                            windowmaterial="glass" windowstyle="framed"
                            windowwidth="3" /><opening><split axis="X"
                            children="nt140350002036184|nt140350002036632
                            " parent="nt140350002035064" splitvalues
                            ="0.3261904761904762,0.6738095238095239" /><
                            elements parent="nt140350002035064" terms="
                            nt140350002036184,nt140350002036632," /></
                            opening><ruleset file="data.xml" /></
                            appearance></town>
                                <town designation="Paris">
                                        <info color="red" dist="0.8" lat
                                            ="48.8534" lon="2.3488"
                                            material="BricksBasic"
                                            nbfloor="1" roofangle="15"
                                            rooftype="flat" />
                                </town>
        </region>
</country>
country designation="Madagascar">
<region designation="Analamanga">
```

```
    <town designation="Antananarivo" />
    <town designation="Antsirabe" />
</region>
<region designation="Nord">
    <town designation="Antsiranana" />
    <town designation="Mahajanga" />
</region>
<appearance><values bdRooftype="mansard" color="black" doorCol="
    brown" floorsize="4" maxFloors="1" maxHeight="5" maxrotation
    ="" minFloors="1" minHeight="5" openingCol="black" proba=""
    roofColor="black" windowfcolor="black" windowheight="tall"
    windowmaterial="glass" windowstyle="simple" windowwidth="2"
    /><ruleset file="data.xml" /></appearance><appearance><values
     bdRooftype="mansard" color="black" doorCol="brown" floorsize
    ="4" maxFloors="1" maxHeight="5" maxrotation="" minFloors="1"
     minHeight="5" openingCol="black" proba="" roofColor="black"
    windowfcolor="black" windowheight="tall" windowmaterial="
    glass" windowstyle="simple" windowwidth="2" /><ruleset file="
    data.xml" /></appearance></country>
</earth>
```

Bibliographie

[1] Daniel G Aliaga, Carlos A Vanegas et Bedrich Benes. « Interactive example-based urban layout synthesis ». In : *ACM transactions on graphics (TOG)* 27.5 (2008), p. 1-10.

[2] Bedrich Benes, Xiaochen Zhou, Pascal Chang et Marie-Paule R Cani. « Urban Brush : Intuitive and Controllable Urban Layout Editing ». In : *The 34th Annual ACM Symposium on User Interface Software and Technology*. 2021, p. 796-814.

[3] J Benner, A Geiger et K Leinemann. « Flexible generation of semantic 3D building models ». In : *Proc of the 1st Intern. Workshop on Next Generation 3D City Models, Gröger/Kolbe (Eds.), Bonn*. 2005, p. 17-22.

[4] Gonzalo Besuievsky et Gustavo Patow. « Recent advances on LoD for procedural urban models ». In : *PROCEEDINGS, IQMULUS 1ST WORKSHOP ON PROCESSING LARGE GEOSPATIAL DATA*. 2014, p. 1.

[5] Filip Biljecki, Hugo Ledoux et Jantien Stoter. « GENERATION OF MULTI-LOD 3D CITY MODELS IN CITYGML WITH THE PROCEDURAL MODELLING ENGINE RANDOM3DCITY. » In : *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 3.1 (2016).

[6] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova et Arzu Çöltekin. « Applications of 3D city models : State of the art review ». In : *ISPRS International Journal of Geo-Information* 4.4 (2015), p. 2842-2889.

[7] H Bunke et X Jiang. *Dreidimensionales Computersehen. Gewinnung und Analyse von Tiefenbildern*. 1997.

[8] Luisa G Caldas et Leslie K Norford. « Shape generation using pareto genetic algorithms : integrating conflicting design objectives in low-energy architecture ». In : *International journal of architectural computing* 1.4 (2003), p. 503-515.

[9] Hau Hing Chau, Xiaojuan Chen, ALISON McKAY et Alan de Pennington. « Evaluation of a 3D shape grammar implementation ». In : *Design computing and cognition'04*. Springer, 2004, p. 357-376.

[10] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller et Eugene Zhang. « Interactive procedural street modeling ». In : *ACM Transactions on Graphics* 27.3 (2008), p. 1. issn : 07300301. doi : 10.1145/1360612.1360702.

[11] Orestes Chouchoulas. « Shape evolution : an algorithmic method for conceptual architectural design combining shape grammars and genetic algorithms ». Thèse de doct. University of Bath, 2003.

[12] Kenneth A De Jong, William M Spears et al. « Using genetic algorithms to solve NP-complete problems. » In : *ICGA*. 1989, p. 124-132.

[13] Ilke Demir, Daniel G Aliaga et Bedrich Benes. « Procedural editing of 3d building point clouds ». In : *Proceedings of the IEEE International Conference on Computer Vision*. 2015, p. 2147-2155.

[14] Ilke Demir, Daniel G Aliaga et Bedrich Benes. « Proceduralization for editing 3d architectural models ». In : *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, p. 194-202.

[15] Ilke Demir, Daniel G Aliaga et Bedrich Benes. « Proceduralization of Urban models ». In : *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE. 2017, p. 1-4.

[16] Christoph Dold et Claus Brenner. « Automatic matching of terrestrial scan data as a basis for the generation of detailed 3d city models ». In : (1996).

[17] Marek Fiser, Bedrich Benes, Jorge Garcia Galicia, Michel Abdul-Massih, Daniel G Aliaga et Vojtech Krs. « Learning geometric graph grammars ». In : *Proceedings of the 32nd spring conference on computer graphics*. 2016, p. 7-15.

[18] William Eric Leifur Grimson, Daniel P Huttenlocher et al. *Object recognition by computer : the role of geometric constraints*. Mit Press, 1990.

[19] Zifeng Guo et Biao Li. « Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system ». In : *Frontiers of Architectural Research* 6.1 (2017), p. 53-62.

[20] J Halatsch, A Kunze et G Schmitt. « Using shape grammars for master planning ». In : *Design Computing and Cognition'08* (2008).

[21] Txomin Hermosilla, Luis A Ruiz, Jorge A Recio et Javier Estornell. « Evaluation of automatic building detection approaches combining high resolution images and LiDAR data ». In : *Remote Sensing* 3.6 (2011), p. 1188-1210.

[22] John H Holland. « Genetic algorithms ». In : *Scientific american* 267.1 (1992), p. 66-73.

[23] Sumbul Khan et Scott C Chase. « Strategic style change using grammar transformations ». In : *AI EDAM-Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 30.4 (2016), p. 488-506.

[24] Thomas H Kolbe. « Representing and exchanging 3D city models with CityGML ». In : *3D geo-information sciences*. Springer, 2009, p. 15-31.

[25] Thomas H Kolbe, Gerhard Gröger et Lutz Plümer. « CityGML : Interoperable access to 3D city models ». In : *Geo-information for disaster management*. Springer, 2005, p. 883-899.

[26] P Koutsourakis, L Simon, O Teboul et G Tziritas. « Single view reconstruction using shape grammars for urban environments ». In : *2009 IEEE 12th* (2009).

[27] Robert G Laycock et AM Day. « Automatically generating roof models from building footprints ». In : (2003).

[28] Thomas Lechner, Ben Watson, Uri Wilensky et Martin Felsen. « Procedural city modeling ». In : *1st Midwestern Graphics Conference* (2003), p. 1-6. url : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.8874.

[29] Jess Martin. « Procedural House Generation : A method for dynamically generating floor plans ». In : *ACM Transactions on Graphics* (2006), p. 1-2. doi : 10.1.1.97.4544.

[30] Jon McCormack. « of L-System Grammars for Computer Graphics Modelling ». In : *Complex Systems : from biology to computation* (1993), p. 118.

[31] P Müller, P Wonka, S Haegler, A Ulmer et L Van Gool. « Procedural modeling of buildings ». In : *Acm Transactions On* (2006).

[32] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer et Luc Van Gool. « Procedural modeling of buildings ». In : *Acm Transactions On Graphics (Tog)*. T. 25. 3. ACM. 2006, p. 614-623.

[33] Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes et Adrien Bousseau. « Interactive sketching of urban procedural models ». In : *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 1-11.

[34] Tiavina Tantely Nivolala, Cédric Sanza, Véronique Gaildrat, Princy Randriambololondrantomalala et Jean-Pierre Jessel. « Buildings modeling : from the shape grammar specification to the IFC model ». In : *Proceedings of the 16th Annual EuroVR Conference - 2019*. 2019, p. 74-89.

[35] Yoav IH Parish et Pascal Müller. « Procedural modeling of cities ». In : *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, p. 301-308.

[36] Victoria Rautenbach, Yvette Bevis, Serena Coetzee et Carin Combrinck. « Evaluating procedural modelling for 3D models of informal settlements in urban design activities ». In : *South African Journal of Science* 111.11-12 (2015), p. 1-10.

[37] Kristina Shea. « eifForm : a generative structural design system ». In : *2000 ACSA Technology Conference : Emerging Technologies and Design : The Intersection of Design and Technology*. 2000, p. 87-91.

[38] M Sinning-Meister, A Gruen et H Dan. « 3D City models for CAAD-supported analysis and design of urban areas ». In : *ISPRS Journal of Photogrammetry and Remote Sensing* 51.4 (1996), p. 196-208.

[39] Alexandra Stadler, Claus Nagel, Gerhard König et Thomas H Kolbe. « Making interoperability persistent : A 3D geo database based on CityGML ». In : *3D Geo-information sciences*. Springer, 2009, p. 175-192.

[40] George Stiny et James Gips. « Shape Grammars and the Generative Specification of Painting and Sculpture. » In : *IFIP Congress (2)*. T. 2. 3. 1971.

[41] George Stiny et William J Mitchell. « The palladian grammar ». In : *Environment and planning B : Planning and design* 5.1 (1978), p. 5-18.

[42] K Ulm. « Virtual 3D City Models-satisfaction through sustainability' ». In : *Geomatics World* 18.6 (2010), p. 16-8.

[43] Carlos A Vanegas, Daniel G Aliaga et Bedrich Benes. « Building reconstruction using manhattan-world grammars ». In : *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, p. 358-365.

[44] Carlos A Vanegas, Daniel G Aliaga, Bedrich Benes et Paul A Waddell. « Interactive design of urban spaces using geometrical and behavioral modeling ». In : *ACM transactions on graphics (TOG)* 28.5 (2009), p. 1-10.

[45] Peter Wonka, Michael Wimmer et William Ribarsky. « Instant Architecture ». In : (2010).

[46] Peter Wonka, Michael Wimmer, François Sillion et William Ribarsky. *Instant architecture*. T. 22. 3. ACM, 2003.

[47] Zhihang Yao, Claus Nagel, Felix Kunde, György Hudra, Philipp Willkomm, Andreas Donaubauer, Thomas Adolphi et Thomas H Kolbe. « 3DCityDB-a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML ». In : *Open Geospatial Data, Software and Standards* 3.1 (2018), p. 1-26.

[48] Xiaowei Zhang, Aly Shehata, Bedrich Benes et Daniel Aliaga. « Automatic deep inference of procedural cities from global-scale spatial data ». In : *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 7.2 (2020), p. 1-28.

[49] Shengchuan Zhou, Innfarn Yoo, Bedrich Benes et Ge Chen. « A hybrid level-of-detail representation for large-scale urban scenes rendering ». In : *Computer Animation and Virtual Worlds* 25.3-4 (2014), p. 243-253.