

POLISH - JAPANESE ACADEMY OF INFORMATION TECHNOLOGY

The Faculty of Information Technology

Ph.D. Dissertation

Generation and Aesthetic Evaluation  
of Architectural Forms  
with the Use of Evolutionary Design

Author:  
Agnieszka Mars

Supervisor:  
Prof. Ewa Grabska

Kraków, 2023

# Acknowledgements

The completion of this dissertation would not have been possible without the people I had the privilege to work with during my research.

First, I would like to thank my advisor Prof. Ewa Grabska for her consistent intellectual guidance during these years. With her insight and advice, she participated actively in every part of this dissertation. I am grateful for introducing me to the field of computational aesthetics and for the hours of inspiring discussions which helped me to clarify the concepts I worked on.

I was lucky to have an opportunity to implement the developed ideas due to the cooperation with DTP-SOFT company. I would like to thank Dariusz Kondys, Artur Sowa and Artur Turkawski who made it possible. I am particularly thankful to my direct collaborators: to Jan Bielański, for his accuracy, precision and deep analysis, to Paweł Mogiła, whose creativity helped to solve unsolvable problems, and to Michał Mogiła, for the pure magic he did as a developer.

I also want to thank my colleagues from The Department of Design and Computer Graphics at the Jagiellonian University in Cracow. Their guidance, insightful comments and willingness to help, contributed to the final results of my research.

Finally, my gratitude goes to my family and friends who made it all easier in many ways. In particular, I would like to thank Paweł for his persistent support and the refreshing perspective on my research, and to Jaśmina, who made sure Mum got enough break from work.

*To my grandfather Lucjan Chołda*

Contents

[Acknowledgements iii](#_Toc137205116)

[Introduction xi](#_Toc137205117)

[Chapter 1. The Visual Representation of Architectural Objects 1](#_Toc137205118)

[1.1. Models of visual perception 3](#_Toc137205119)

[1.1.1. Recognition-by-components (RBC) 4](#_Toc137205120)

[1.1.2. Application of RBC in the visual representation of architecture 7](#_Toc137205121)

[Chapter 2. The Graph Representation of Architectural Objects 15](#_Toc137205122)

[1.2. Graphs 15](#_Toc137205123)

[1.3. Graph grammars 25](#_Toc137205124)

[Chapter 3. Generative Procedures 30](#_Toc137205125)

[1.4. Cellular automata 30](#_Toc137205126)

[1.5. Genetic algorithms 31](#_Toc137205127)

[1.6. L-systems 31](#_Toc137205128)

[1.7. Swarm intelligence 32](#_Toc137205129)

[1.8. Shape grammars 32](#_Toc137205130)

[1.9. Graph grammars 34](#_Toc137205131)

[1.10. Artificial neural networks 34](#_Toc137205132)

[1.11. Other approaches 35](#_Toc137205133)

[Chapter 4. Genetic Algorithms in Creative Design 37](#_Toc137205134)

[1.12. Creative design 37](#_Toc137205135)

[1.12.1. Creativity 37](#_Toc137205136)

[1.12.2. Design 38](#_Toc137205137)

[1.12.3. Creative design 39](#_Toc137205138)

[1.13. Genetic algorithms in design 40](#_Toc137205139)

[1.13.1. Evolutionary design 40](#_Toc137205140)

[1.13.2. The genetic algorithm 41](#_Toc137205141)

[Chapter 5. Computational Aesthetics 45](#_Toc137205142)

[1.14. Definition of computational aesthetics 45](#_Toc137205143)

[1.15. Measuring aesthetics 46](#_Toc137205144)

[1.15.1. Birkhoff aesthetic measure 46](#_Toc137205145)

[1.15.2. Zipf’s law 48](#_Toc137205146)

[1.15.3. Machine learning approach 48](#_Toc137205147)

[1.16. Aesthetics of architecture 49](#_Toc137205148)

[1.16.1. Proportion ratios 49](#_Toc137205149)

[1.16.2. Fractal dimension 51](#_Toc137205150)

[1.16.3. Gestalt perception laws 53](#_Toc137205151)

[1.16.4. Aesthetical components of the architecture 54](#_Toc137205152)

[Chapter 6. Application of Computational Aesthetics in Generative Architecture Design 57](#_Toc137205153)

[1.17. Aesthetics in generation 57](#_Toc137205154)

[1.17.1. Component types 58](#_Toc137205155)

[1.17.2. Relations between components 62](#_Toc137205156)

[1.17.3. Analogical rules 62](#_Toc137205157)

[1.17.4. Reverted rules 64](#_Toc137205158)

[1.17.5. Patterns 65](#_Toc137205159)

[1.17.6. Number of components 67](#_Toc137205160)

[1.17.7. Multiple reference buildings 69](#_Toc137205161)

[1.18. Evaluation of aesthetics 70](#_Toc137205162)

[1.18.1. Alignment 71](#_Toc137205163)

[1.18.2. Mirror symmetry 72](#_Toc137205164)

[1.18.3. Rotational symmetry 73](#_Toc137205165)

[1.18.1. Stability 74](#_Toc137205166)

[1.18.2. Number of components 75](#_Toc137205167)

[1.18.3. Number of component types 76](#_Toc137205168)

[1.18.4. Aesthetic measure for comparison of architectural forms 77](#_Toc137205169)

[Chapter 7. The Genetic Algorithm for Generative Design of Architectural Objects in a Given Aesthetic Character 80](#_Toc137205170)

[1.19. The individual 80](#_Toc137205171)

[1.20. The phenotype 80](#_Toc137205172)

[1.21. The initial population 81](#_Toc137205173)

[1.21.1. The graph representation of the reference buildings 81](#_Toc137205174)

[1.21.2. Constructing the graph grammar 84](#_Toc137205175)

[1.21.3. Generating the initial population 91](#_Toc137205176)

[1.22. The genotype 94](#_Toc137205177)

[1.23. The fitness function 97](#_Toc137205178)

[1.24. The selection 97](#_Toc137205179)

[1.25. The crossover 97](#_Toc137205180)

[1.26. The mutation 100](#_Toc137205181)

[1.27. The termination condition 102](#_Toc137205182)

[Chapter 8. The Results 103](#_Toc137205183)

[1.28. Examples of program results 103](#_Toc137205184)

[1.29. Summary 111](#_Toc137205185)

[1.30. Further work 112](#_Toc137205186)

[Appendices 113](#_Toc137205187)

[Appendix A: The algorithms for determining the number of order relations 114](#_Toc137205188)

[The algorithm for determining the number of alignment relations in a group of components 114](#_Toc137205189)

[The algorithm for determining the number of partial mirror symmetries in a group of components 116](#_Toc137205190)

[The algorithm for determining the number of partial rotational symmetries in a group of components 117](#_Toc137205191)

[Appendix B: The source code of the genetic algorithm 120](#_Toc137205192)

[List of Figures 123](#_Toc137205193)

[Bibliography 127](#_Toc137205194)

# Introduction

The automatic generation of designs has been a subject of research for decades. The following dissertation aims to contribute to this area of computer-aided design. Its main objective is to propose an approach to equip a generative system with some imitation of the human sense of aesthetics, which, despite its inevitable limitations, will result in producing designs of given aesthetical characteristics. The presented approach considers the generation of architectural forms on the example of virtual cities, however, it can be extended to other applications of automatic generation of designs.

The state of the art in the research regarding computational aesthetics and generative design is discussed. The approaches to measuring the aesthetics of generated designs are presented [1], as well as an overview of generative tools with selected examples. Genetic algorithms are described in the context of their applicability in creative design.

The model of visual perception developed by I. Biederman [2] is used to construct an alphabet of geometric primitives that can be widely used in designing artifacts, especially architectural forms. The simplicity and versatility of Biederman's model allow for organizing design spaces in a way that corresponds to human perception. Referring to human perception seems necessary to make design spaces explorable for solutions that meet certain aesthetic criteria, as the sense of aesthetics is specific to humans.

The internal representation of the designs is performed by graphs. Labeled attributed composition graphs (CP-graphs) [3] are proposed to denote the design's components and the relations between them. Once the graph representation of the reference building is created, it can be used to construct a graph grammar. For this purpose, simple CP-graph grammars are proposed. A simple CP-graph grammar constructed based on the reference building's internal representation generates a language composed of graphs that represent designs. These designs are supposed to have the characteristics of the reference building, however, more validation is necessary to reject designs that do not fulfill basic aesthetical or functional criteria.

The task of validation and exploration of design spaces for the most optimal solutions is performed by a genetic algorithm [4], which is proposed as a generative tool. The genetic algorithm presented in this dissertation uses phenotypes based on Biederman's perception model and genotypes in the form of a sequence of simple CP-graph grammar rules. The fitness function of the genetic algorithm evaluates the design's aesthetical coherence with the reference building, as well as some basic aesthetical and functional properties.

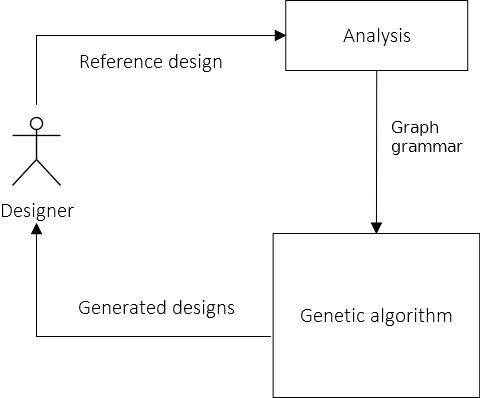


Figure . Schema of the generative system

Figure 1 shows a schematic picture of the approach to the generative design presented in this dissertation. A designer is required to provide a model of the reference building, which should be created as a configuration of predefined primitives. The model is then analyzed – its internal representation in the form of a graph is automatically created and used to construct a graph grammar. This is the moment when the genetic algorithm starts its work. The graph grammar is used to generate the initial population of designs, which are then evolved to obtain satisfactory results. The generated designs with aesthetical characteristics of the reference design are finally provided to the designer.

The presented system has been implemented by DTP-SOFT company in Rabka-Zdrój, Poland, as a part of the project entitled “The City Creator”. The project, financed by the National Center for Research and Development and developed in the years 2019-2021, aimed to provide a universal solution for generating virtual cities of user-defined characteristics. It consisted of four stages. In the first stage, the methods of structural analysis of buildings and defining characteristics of architectural forms were developed. The second stage regarded the generation and selection of derivative buildings and placing them in the virtual scenery. The third stage was focused on the analysis and the generation of textures, while during the last one, the tests and system optimization were performed as well as the implementation of the graphical user interface.

The author’s contribution to "The City Creator" project as an expert on procedural generation algorithms was to develop the concepts required in the analysis and synthesis of architectural forms and implement the genetic algorithm, whose most important module regarded aesthetic evaluation of buildings. This task has been possible thanks to the project supervisor Professor Ewa Grabska, who is the author of composition graphs adapted as an internal representation of our generative system, as well as the hard conceptual and programming work of the project managers Artur Sowa and Artur Turkawski, and my colleagues Jan Bielański, Michał Mogiła, and Paweł Mogiła, who provided, among many others, the code of visualization and internal representation of buildings, the implementation of graph grammars and the graphical user interface. The system has been implemented as a Blender add-on.

The concepts developed as the author’s contribution to the project have resulted from several years of research [5–10] and are presented in this dissertation. They include:

1. Using a design structure of the reference building as a model of the designer style, with an assumption that: "By a style of buildings, we will understand a rule relating to a class of buildings for which its components can be structured by using the same set of aesthetic criteria" [11]. The design structure is characterized by a set of components and relations between them, while the designer style determines the aesthetic character of the designs;
2. The application of Biederman’s volumetric representation for the purpose of the computational aesthetics;
3. Simple CP-grammars: context-free graph grammars used for generating graphs that represent designs and enable the application of design patterns;
4. Identification of aesthetic components in the generation and evaluation of designs;
5. The aesthetic measure for the comparison of aesthetics of different designs;
6. The construction of the genetic algorithm generating designs in a given character, particularly:
   1. generation if the initial population based on the reference design,
   2. using genotype in the form of a sequence of graph grammar rules that corresponds to the binary encoding and enables simple crossover and mutation operators,
   3. the fitness function based on a comparison of aesthetic qualities between the reference and derivative design.

The first chapter, “Computational Aesthetics and Generative Procedures" presents state of art and related work with an emphasis on the computational aesthetics of architectural forms. The second chapter, "Genetic Algorithms in Creative Design", explains such ideas as creativity, design, and creative design, and describes the functioning of genetic algorithms. The next chapters refer to the implemented genetic algorithm. Firstly, the visual representation of architectural forms based on Biederman’s model of visual perception is described in the chapter “The Visual Representation of Architectural Objects”. This approach is used to construct phenotypes of individuals in the population, where each individual is a single building. The chapter entitled “The Graph Representation of Architectural Objects” presents graphs used for the internal representation of buildings and graph grammars used for generating the initial population. Based on graph grammar rules, genotypes are constructed. In the chapter “The Evaluation of Architectural Objects” the fitness function of the genetic algorithm is proposed. It combines the ideas of aesthetic evaluation of artificially created designs presented in the first chapter with the basic functional requirements of architectural forms. The next chapter, “The Genetic Algorithm for Generative Design of Architectural Objects in a Given Aesthetic Character” presents the functioning of the applied genetic algorithm with the use of the concepts introduced before. Besides the phenotype and genotype representation and the fitness function based on the ideas from the previous chapters, the genetic operators – the mutation and the crossover – are described, as well as the selection process. Finally, in the last chapter “The Results”, examples of resultant solutions are presented and some conclusion is made.

# Chapter 1. The Visual Representation of Architectural Objects

As human interaction with the environment relies mainly on visual information, visualization is probably the most intuitive method of communicating design concepts. Sketching and drawing enables people not only to share their ideas with others but also to perform a sort of dialogue with themselves during the process of conceptualization. Internal mental representation is externalized and internalized again through the act of visual perception. This gives the designer an opportunity to refresh and modify their ideas [12].

Visualization is also indispensable in human-computer interaction. Every task of computer-aided design (CAD) requires adequate visual representation. Marr [13] gives the following definition of representation: „A representation is a formal system for making explicit certain entities or types of information, together with a specification of how the system does this. And I shall call the result of using a representation to describe a given entity a description of the entity in that representation.”

To implement visual representation in a CAD system one needs to determine a set of primitives corresponding to the parts of the design objects and a set of relations between the primitives corresponding to the relations between the design objects' parts. Architectural design requires primitives symbolizing buildings or building components. The primitives and relations between them strictly depend on the design task – for instance, a floor plan is represented with a different set of graphic components than a city map. Figure 2 presents two alternative representations of an apartment designed in the program Floorplanner [14]. The top image contains a floor plan, while in the bottom one, we can see 3D visualization of the apartment’s interior. In each case, different sets of primitives were used for representation.



Figure . Alternative representations of an apartment

In the case of automatic generation of designs, it is essential for the computer program to evaluate the obtained results. Automatic evaluation of architectural forms is a difficult task since aesthetics – a very subjective and hard-to-define criterium – is one of the most important factors that should be taken into account. To approach the goal of computational aesthetic measure for computer-aided design of architecture this thesis proposes to use a model of visual perception for the representation of architectural forms. Such an approach enables computer programs to imitate to some degree human cognitive processes occurring during object recognition and evaluation.

## Models of visual perception

Visual perception is the most common way in which the human brain performs recognition. An object is perceived, analyzed, and finally identified. There are two main approaches to finding an explanation of how exactly this process is conducted [15]. The first one assumes that visual perception is view-dependent, i.e., a perceived object is compared to multiple views of different objects stored in memory. The most accurate match is found, which enables object identification. Figure 3 presents several views of a hare. Seeing a hare activates an area of visual memory in which the most adequate of hare views is stored. This concept was developed among others by Bülthoff and Edelman [16], Tarr [17], Perrett et al. [18], and Riesenhuber and Poggio [19].

Another approach to visual perception is view-invariant. This means that object recognition is performed by the extraction of basic components a perceived object is built from. Such components can be identified independently of the viewpoint. Investigating their configuration enables recognition. This approach was presented by Marr & Nishihara [20].

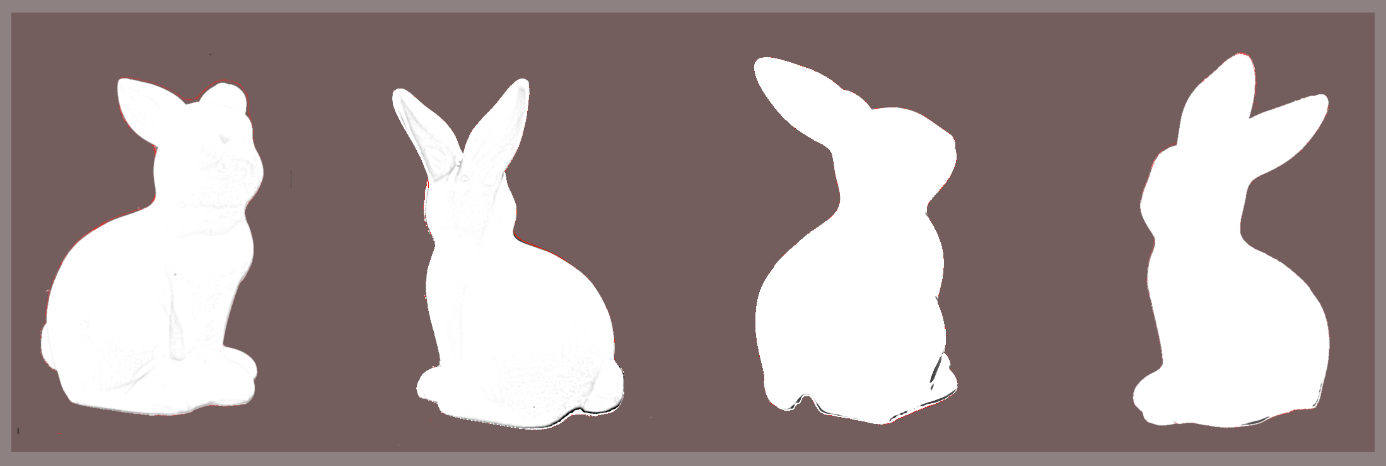


Figure . Different views of an object

### Recognition-by-components (RBC)

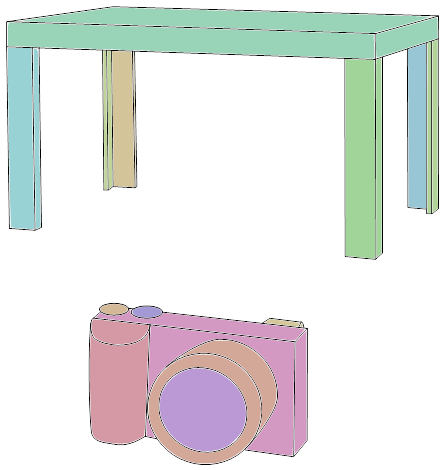


Figure . Objects composed from geons

The view-invariant model of visual perception has been further developed by Biederman in his recognition-by-components (RBC) theory [2]. According to the RBC theory, the human brain perceives most objects as a composition of some idealized solids called geons. Figure 4 illustrates the division of some objects into such components (marked with different colors). Geons are extracted during the perception process in the areas of sharp concavities. They are formed by running a cross-section along an axis. Four main properties characterize geons:

1. cross-section edge (**S** for polygons or **C** for other shapes),
2. cross-section symmetry (**++** for a figure invariant under reflection and 90º rotations, **+** for a figure invariant under reflection only and **-** for other figures),
3. cross-section size (constant **++**, contracted **-** or expanded and contracted **--**)
4. and axis (straight **+** or curved **-**).

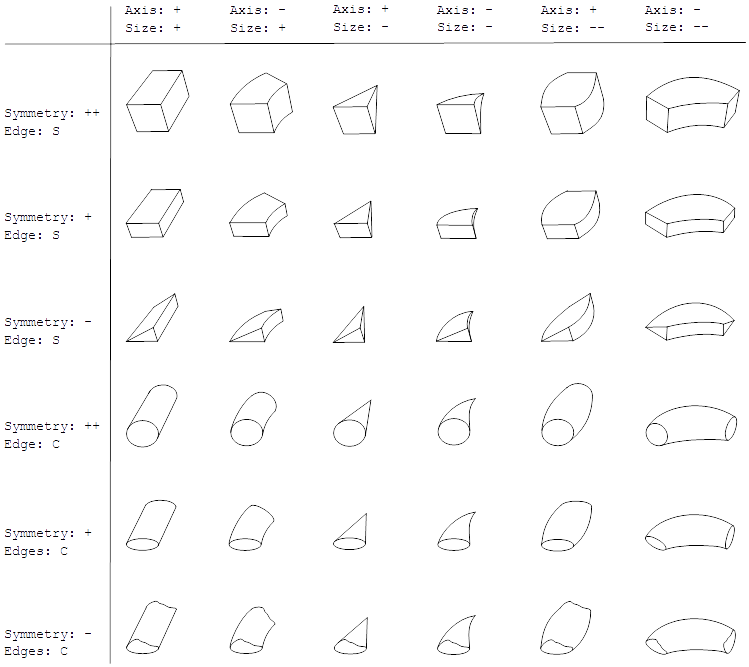
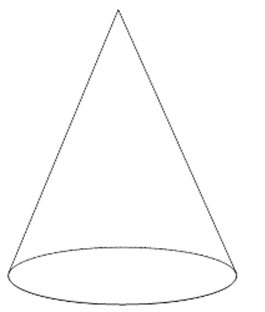


Figure . Thirty-six geons as combinations of four non-accidental properties

These properties are non-accidental, which means they can be quickly and accurately recognized independently of the point of view. Combinations of the four main non-accidental properties give thirty-six different geons (see Figure 5). The RBC theory assumes the existence of other non-accidental properties, like figure termination (e.g., pointed or truncated, as in Figure 6 a) and b), respectively, which may result in multiple subtypes of each geon.



a)

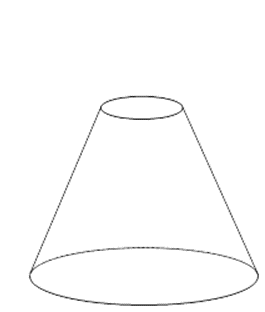
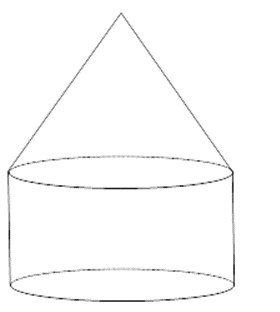
  
b)

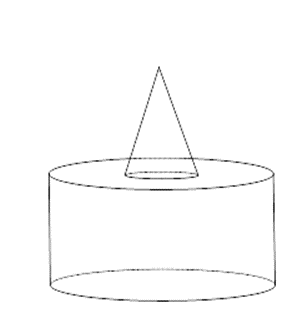
Figure . Different types of geon termination

Besides the mentioned non-accidental properties there is plenty of metric properties used to define a solid, like width, height or angles between edges. The cost of perception of metric properties is relatively high – it takes a long time to recognize them, and the result is prone to errors. This is why it is assumed that metric properties are not crucial for object recognition. Biederman's geons are therefore abstract entities defined by non-accidental properties. During the process of visual perception, a perceived object is divided into parts (actual solids defined by multiple properties), each of them matched to the most accurate geon (an idealized solid defined by non-accidental properties only).

To identify an object, the observer's brain needs not only to recognize geons but also the relations between them. There are two main types of spatial relations: end-to-end (presented in Figure 7 a)), when one geon prolongs another one, and end-to-side (Figure 7 b)), when one geon is attached to the surface of another one. Geons' surfaces are important for image understanding, however, it is enough to determine whether components are attached with a top basis, a bottom basis, or a side surface (and in the case of different side surfaces, its generalized size, e.g., small or large).

****

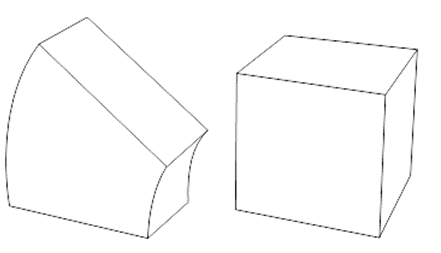
a)

****

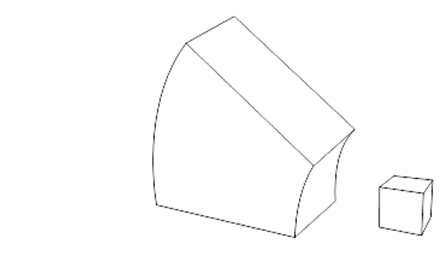
b)

Figure . Spatial relations between geons

Despite spatial relations between geons, there are also size relations. According to the RBC theory, there are two types of size relation: when one geon is significantly larger than another one (as in Figure 8 a)) and when they are more or less the same size (Figure 8 b)).



a)



b)

Figure . Size relations between geons

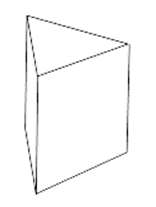
### Application of RBC in the visual representation of architecture

View-independent approach to object recognition seems useful for creating computational models of visual perception. The RBC theory can be relatively easily implemented in automatic vision. One of the examples has been published in [21], where object segmentation is achieved with a set of geons that was reduced to symmetrical solids (symgeons) to eliminate ambiguities. In CAD, the application of geons is even more intuitive, as many computer programs allow users to compose complex shapes from pre-defined primitives. Architecture design with the use of the RBC model has been described in [12].

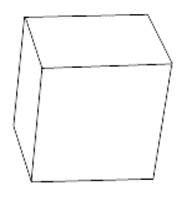
This dissertation adapts the Biederman theory of geons for a visual representation of buildings in the automatic generation of architectural forms. The next chapters contain modifications and extensions of the RBC model that enable its implementation.

#### Components

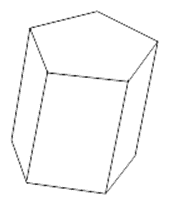
The proposed approach to the visual representation of architectural objects uses Bederman's geons as a basis for determining a set of components that construct most buildings. As in the RBC model, each component is a figure drawn by extruding a two-dimensional shape along an axis. Combinations of the four **non-accidental properties** (axis, cross-section edge, cross-section symmetry, and cross-section size) define **component types**. Additional non-accidental properties (**detailed properties**), i.e., cross-section edge number, cross-section curvature, axis curvature, the function of cross-section size change, and the ranges of component width, length and height are specific for a given component type and define **component subtypes**. **Metric properties** (location, rotation, vertices, etc.) are required to define **components**, which are instances of component types.



a)



b)



c)

Figure . Different values of the cross-section edge number property

Figure 9 presents three components of the same non-accidental properties (axis: +, cross-section edge: S, cross-section symmetry: +, cross-section size: ++) that have a different value of a detailed property cross-section edge number: 3 in Figure 9 a), 4 in Figure 9 b) and 5 in Figure 9 c). The following list presents all the non-accidental attributes and their values:

1. Main non-accidental properties (based on the RBC model):
   1. cross-section edge (**S** for polygons or **C** for other shapes),
   2. cross-section symmetry (**++** for a figure invariant under reflection and 90º rotations, **+** for a figure invariant under reflection only and **-** for other figures),
   3. cross-section size (constant **+** or variant -, which unifies two values from the RBC model: contracted - and expanded and contracted --),
   4. axis (straight **+** or curved **-**).
2. Detailed properties:
   1. edge number (natural number greater or equal to three),
   2. cross-section curvature,
   3. axis curvature,
   4. function of cross-section size change,
   5. range of component width,
   6. range of component length,
   7. range of component height.

Figure 10 shows detailed properties used to define subtypes of each geon type.



Figure . Detailed properties of each component type

#### Curves

The proposed component types contain three detailed properties describing curvatures: cross-section curvature, axis curvature, and curvature of cross-section size change. For components, each curvature may be presented by NURBS (Non-Uniform Rational B-Spline). The NURBS curves are defined by control points that form vertices of a polyline, as shown in Figure 11. Figure 12 presents an application of the NURBS curves in defining components.

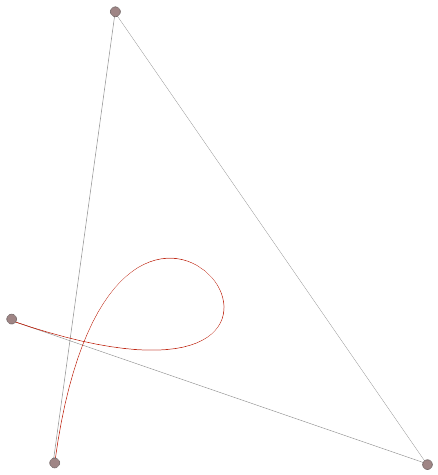


Figure . Example of a NURBS curve

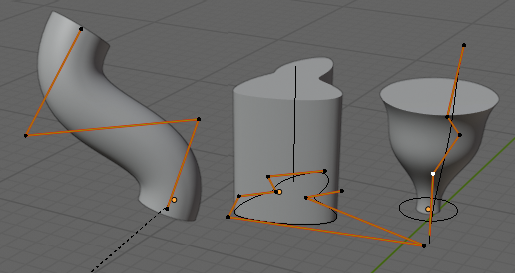


Figure . Control polylines of component curves

#### Relations

As mentioned above, the RBC model offers two types of relations between geons: end-to-end and end-to-side. However, the computational representation of architectural forms in design seems to require more precise specification of relations. It is important to determine in which surfaces' areas two solids are joined – for example, they might be centered or aligned to one's edges.

The following relations between components have been proposed:

1. end-to-end, in which connected surfaces overlap, as presented in Figure 13,
2. end-to-side, in which connected surfaces do not overlap. This case can be further divided regarding areas of connected surfaces, for instance:
   * center-to-center (Figure 14), in which connected surfaces do not overlap, but their central points overlap,
   * edge-to-edge (Figure 15), in which edges of the connected surfaces overlap,
   * irregular end-to-side (Figure 16), in which connected surfaces do not overlap and no relation center-to-center and edge-to-edge occur.

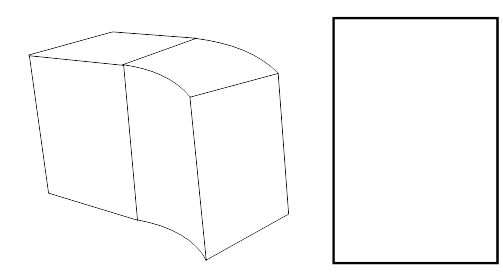


Figure . End-to-end relation between components

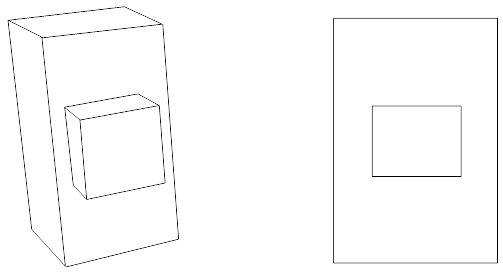


Figure . Center-to-center relation between components

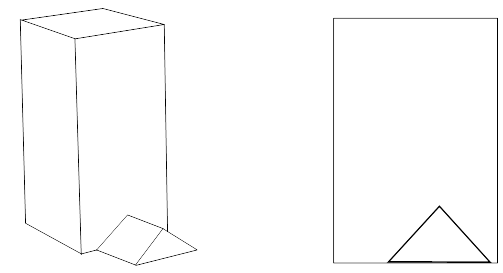


Figure . Edge-to-edge relation between components

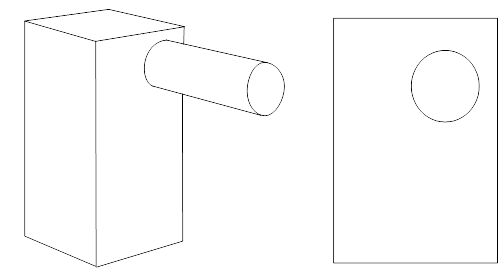


Figure . End-to-side relation between components

Figures 13 - 16 present relations described in 1, 2 a), 2 b) and 2 c), respectively. Each picture contains a sketch of the relation example and a projection of the connected surfaces.

# Chapter 2. The Graph Representation of Architectural Objects

To generate design objects and perform operations on them, generative systems need internal representations. These are data structures that store information about designs (as object's components and their properties) and may be transformed automatically by a computer program to obtain results that satisfy the designer's requirements. Based on the internal representation, a generative system is able to prepare a visualization of an object.

A common tool for internal representation in CAD is graphs [22]. In the graph knowledge representation, the way of organization, processing and manipulation of knowledge is based on the spatial relations between artifacts' components. Such an approach corresponds with the RBC theory, in which object recognition depends on the types and configuration of basic components (geons). This suggests the way of defining an artifact structure as a graph with nodes representing components interconnected by edges that represent relationships between them.

The generative system presented in this dissertation aims to create buildings of similar aesthetic characteristic as the one given by a designer as an example (i.e., the reference building). Therefore, the first step to obtain this goal is to create an internal representation of the provided reference building.

## Graphs

Let us recall the basic definition of a labeled directed graph.

**Definition 1.** Let be an alphabet of labels. By a **labeled directed graph** over we understand a tuple , where:

* and are pairwise disjoint sets, whose elements are called nodes and edges, respectively;
* are functions assigning to edges source and target nodes, respectively;
* is an edge and node labeling function.

Figure 17 presents an example of a design object, while Figure 18 shows its internal representation in the form of a labeled directed graph. The graph nodes represent the object's components and are labeled according to their role in design: “block”, “tower” and “roof”. The directed edges represent spatial relations between the components based on the RBC model: “end-to-end” and “end-to-side”.



Figure . A design object

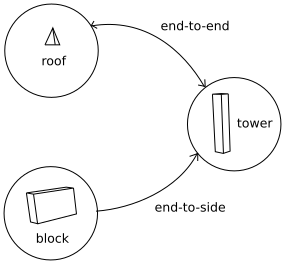


Figure . A labeled directed graph representing an object from Figure 17

It is necessary to note that the way distinct object components are attached plays an essential role in the RBC theory. Modification of one attachment may influence the object identification, as shown in Figure 19. The object is perceived differently depending on the component's surfaces that take part in the attachment relation. Figure 20 presents a labeled directed graph, which represents each of these three objects and disregards differences between them. Therefore, we need a specific type of graph that will be able to identify attachments between components more precisely. Let us consider composition graphs (CP-graphs) [2] for the representation of objects' structures regarding components’ surfaces. Intuitively, a CP-graph is a graph where nodes have explicit connection elements called *bonds*, and edges are connected to bonds.

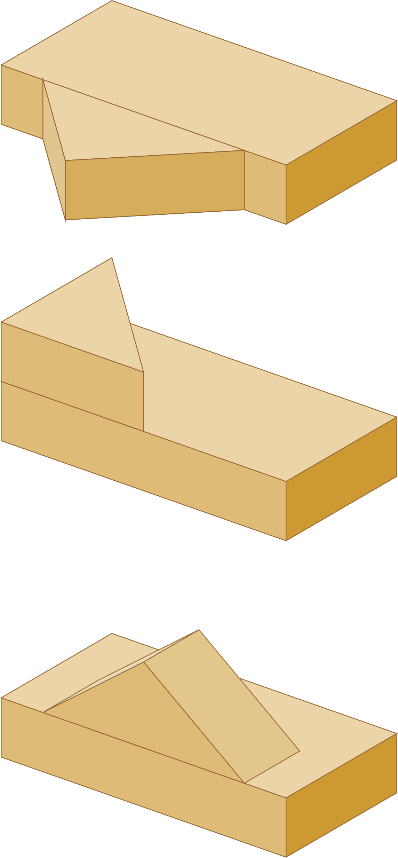


Figure . Different attachments between the same components

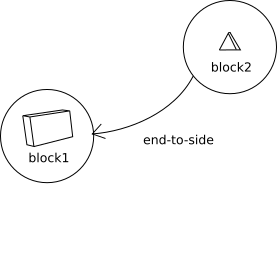


Figure . A labeled directed graph representing the objects from Figure 19

The definition of a CP-graph presented in [23] has been adapted to better correspond to the problems stated in this dissertation.

Let be an alphabet of labels.

**Definition 2.** A **labeled composition graph (labeled CP-graph)** over is a tuple , where:

* , , are pairwise disjoint sets, whose elements are called nodes, edges and bonds, respectively;
* is a function such that defines a sequence of different bonds for each node and , , i.e., each bond is assigned to exactly one node;
* are functions of which at least one is not surjective that assign to edges source and target bond nodes, respectively, in such a way, that ;[3]
* is an edge, node and bond labeling function.

CP-graphs were proposed as a basic computer model representation during the design process [3]. They are graphs where nodes are equipped with explicit connection elements called bonds. An edge labeled with the name of the relation is attached to the bonds corresponding to the arguments (or having attribute values being the arguments) of this relation at the level of the labeled CP-graph. Non-symmetrical relations are represented by a directed CP-graph equipped with two types of bonds: source bonds and target bonds. The CP-graph bonds that are neither source nor target bonds are called *free* bonds and represent potential connections at the higher level hierarchy.

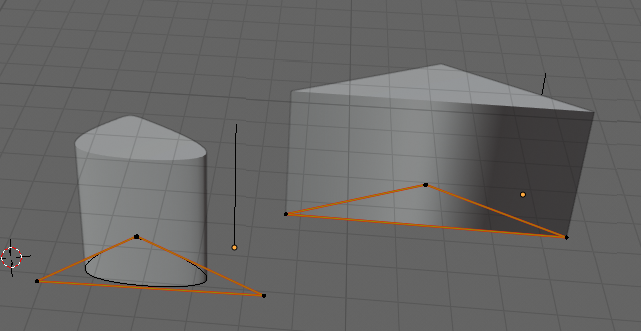


Figure . Two components with different cross section types – a curved one and a triangular one



Figure . CP-graph nodes representing components from Figure 21

The CP-graph bonds represent the component's surfaces – two bases and side surfaces. In the case of a curved cross-section, the side surface bonds are determined by a cross-section control polyline. Figure 21 presents two components with different cross-section types – a curved one and a triangular one. Because the control polyline of the curved cross-section is triangular as well, the number of bonds is the same for both component nodes, which is shown in Figure 22.

CP-graph representation is able to distinguish between the three objects shown in Figure 19. The surfaces of the component labeled as “block1” are represented as follows: the two largest surfaces are labeled as “surface 1” and “surface 2” and indexed with 1 and 2, respectively. The medium surfaces, labeled with “surface 3” and “surface 5”, have indexes 3 and 5, and the smallest ones – “surface 4” and “surface 6” – 4 and 6. The two largest surfaces of the component “block2”, i.e., “surface 1” and “surface 2”, have indexes 1 and 2, the medium one, “surface 3”, is indexed with 3 and the remaining ones, “surface 4” and “surface 5” are indexed with 4 and 5, respectively. Figure 23 shows three different CP-graphs of the buildings from Figure 19.

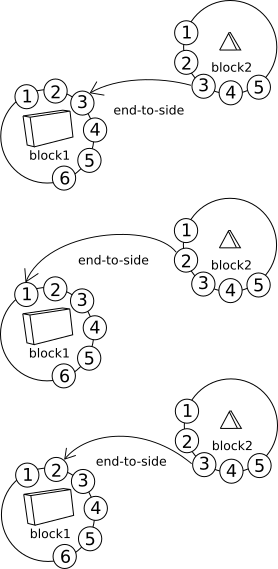


Figure . Labeled directed CP-graphs representing the buildings from Figure 19

Introducing bonds has enabled to distinguish between graph structures of the three objects from Figure 19, which was impossible with the use of a labeled directed graph presented in Figure 20.

Figure 9 contains another CP-graph. It is the representation of the object from Figure 1. The labeled directed graph from Figure 2 has been extended by bonds attached to each node. Again, the bonds represent the component's surfaces and are labeled with surface numbers (“surface 1” for a bottom basis, “surface 2” for a top basis – which in the case of the roof component is reduced to one point – and other numbers for side surfaces), and indexed with corresponding numbers. The graph edges connecting bonds indicate which surfaces take part in each spatial relation.

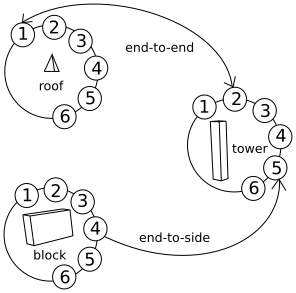


Figure . A CP-graph representing the object from Figure 17

A common solution used in computer-aided design is to allow the user to group design elements. In the presented approach, such groups are treated as CP-graph patterns with the possibility of their repeated use. Among free bonds of a pattern, the **external** bonds are chosen for connecting the pattern with another CP-graph. All the remaining bonds of a pattern are called **internal**. Patterns can be then treated as hierarchical nodes, where only their external bonds can be source and target bonds for edges.

Let us denote elements of a CP-graph  by and by  the set of all labeled CP-graphs over

Let us define for a given CP-graph  the sets  and  which determine the sets of internal bonds and set of external bonds, respectively.

.

**Definition 3.** Let . By a CP-graph **pattern object** for C, we understand a tuple  ), where:

* is a node called a hierarchical node that is not an element of ,
* is a label of that is not an element of

* :{w} is a function that defines a sequence of different bonds for
* CP-graphis a descendant of .

.

In other words, a CP-graph pattern object for is a hierarchical labeled one-node CP-graph with its descendant being and with the set of bonds equal to .

The pattern objects will be further referred to as **hierarchical nodes**, opposite to the other nodes of a CP-graph called **single nodes**.

Figure 25 presents a design object whose components have been grouped into a design pattern called “TOP”. Figure 26 shows a graph representing this building.



Figure . Example of a building with a user-defined pattern

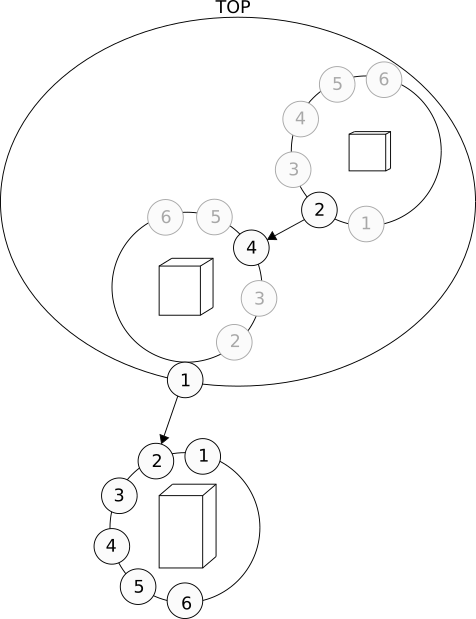


Figure . Graph representing the building from Figure 25

In the presented usage scenario, CP-graphs give information about the object's structure: components, components' surfaces and relations between them. This is, however, not sufficient to fully represent a building design, as information about size, cross-section, symmetry and other properties is often required during the design process. This problem can be solved by adding attributes to graph elements.

Let , , and be sets of node, bond and edge attributes, respectively. Let , where is a set of non-accidental attributes and is a set of metric attributes. Let be a set of attributes for nodes, bonds and edges, such that , denotes its domain, i.e., a set of all possible values of this attribute.

**Definition 3**. An **attributed labeled composition graph (attributed labeled CP-graph)** over is a tuple , where:

* is a labeled CP-graph,
* , and are node, bond and edge attributing functions, respectively.

Pattern objects may be formed by attributed labeled CP-graphs.

Attributed labeled CP-graphs will be further named as CP-graphs. The elements of a CP-graph will be referred to with indexes, e.g., a CP-graph will be described as a tuple .

By a **type of a CP-graph node**, we understand the number of bonds assigned to this node.

Let and be CP-graphs over .

**Definition 4**. The mapping is called **CP-graph homomorphism**, which is denoted as , if:

* and and ,
* , = ) and nodes and are of the same type,
* and ,
* and ), for every ,
* and ), for every ,
* , for every .

CP-graph homomorphism is said to be **isomorphism** if is a bijection. In this case, the CP-graphs and are called **isomorphic** and we write then .

The above definition of CP-graph isomorphism can be extended into pattern objects for CP-graphs.

The isomorphism of CP-graphs descendant of hierarchical nodes and results in the isomorphism of and , providing their labels are identical and they contain the same number of external bonds with the same labels.

Adding information about attribute values to a CP-graph produces a CP-graph instance. It is worth noting that one CP-graph can be the basis for many instances that differ only in the values of the attributes.

**Definition 5**. An **instance of a CP-graph** over is a tuple , where:

* is a CP-graph,
* is a function assigning values to node attributes, where such that,
* is a function assigning values to bond attributes, where such that,
* is a function assigning values to edge attributes, where such that.

The values of a CP-graph instance will be referred to with indexes, e.g., the values of a CP-graph instance will be referred to as , and . The elements of the instantiated CP-graph will also be referred to with indexes, e.g., the instantiated elements of the CP-graph in will be described as a tuple .

Let be an instance of a CP-graph over .

Let us call a **CP-node instance**  resulting from assigning attribute values to a node .

By a **type of a CP-graph node instance**, we understand a pair , where is the number of bonds assigned to this node and is a set of pairs in which and is a value assigned to the attribute .

## Graph grammars

**Definition 8**. A **simple CP-grammar rule**  over is a pair , where:

1. is a CP-graph instance over consisting of one single node or one hierarchical node ,
2. is a CP-graph instance over having two node instances and connected with an edge, where node instance corresponding to is of the same type as , and is an instance of a single node or a hierarchical node.

Let be an instance of a CP-graph over . Application of in consists of the following steps. At first, a condition necessary for the production application is checked, i.e., a node instance is searched in such that is of the same type as . Then if such a node instance exists, a node instance isomorphic to is added to along with a set of edges connecting and such that for each edge , edge is added to such that . Analogically, for each edge , edge is added to such that . Finally, the values of the bond attributes as well as of the node metric attributes of the new components of are assigned.

**Definition 9**. A **simple CP-grammar** over is a pair , where:

1. is a finite set of simple CP-grammar rules over ,
2. is a set of instances of CP-graphs over , each containing at least one node, and is called a set of axioms of G.

Let be a simple CP-grammar.

Let , be CP-graph instances over .

**Definition 10.** We say that is **directly derivable** from in , which is denoted as , if there exists a rule in such that is possible to apply in and the CP-graph resulting from the application of in is isomorphic to .

If there exists a sequence of rules in such that and , we say that is **derivable** from , which is denoted as . The relation is then the reflexive-transitive closure of the binary relation .

**Definition 11.** A **language** generated by is defined as , where .

Figure 28 contains derivation performed by means of this grammar. The final graph presented in the picture refers to the class of graph instances, as the metric attributes assigned during the application of each rule in the sequence may be different in each case of the derivation. Figure 29 shows three buildings represented by the obtained graph. All of them have the same structure, but they differ in some metric properties of the components, like width and height.

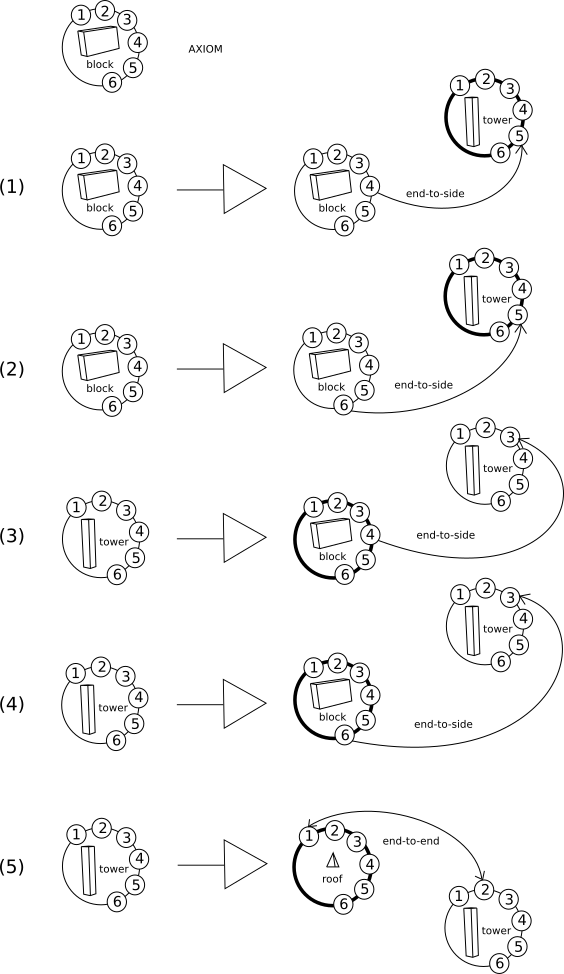


Figure . Example of a simple CP-graph grammar

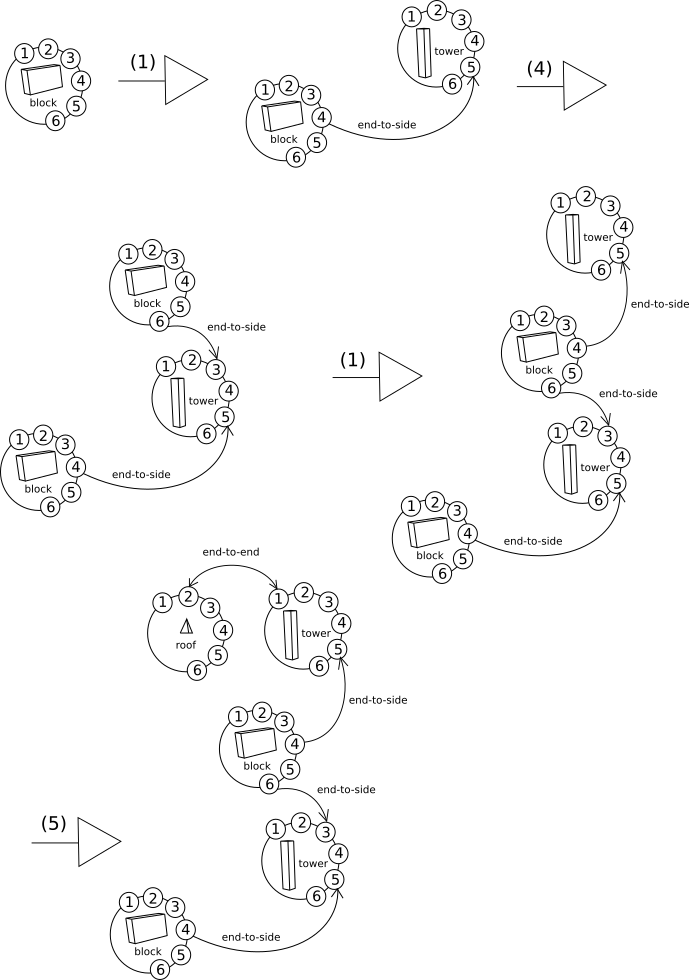


Figure . Example of derivation by means of the grammar shown in Figure 27

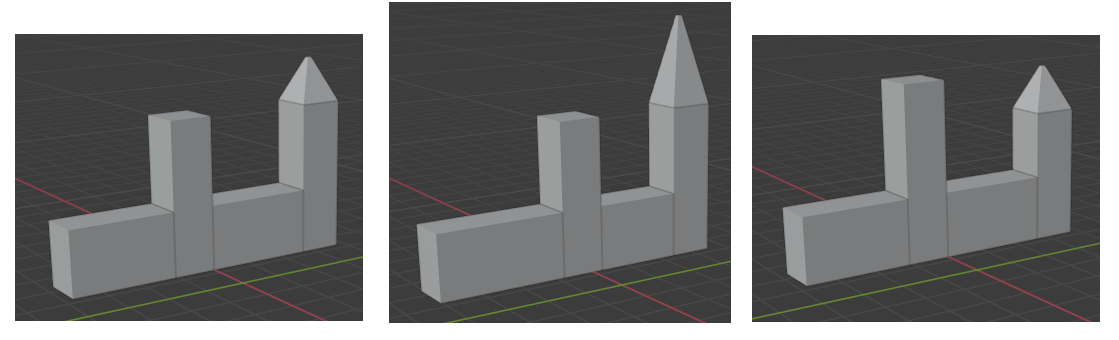


Figure . Buildings represented by the graph derived in Figure 28

# Chapter 3. Generative Procedures

There is a large number of generative systems that aim to develop aesthetically pleasing designs and art. These tools are often equipped with some knowledge about aesthetics, like rules of mixing colors or combining notes. A description and comparison of generative design methods are presented in [24]. The following summary presents the most important of them.

## Cellular automata

A cellular automaton is a mathematical model that consists of a grid and a set of rules that modify its cells based on their neighborhood. The state of the automaton is specified by the values of discrete variables assigned to each cell, which are, in discrete time steps, simultaneously modified according to the defined rules [25]. Cellular automata were discovered by John von Neumann [26] and since then widely used in many domains, such as biology, physics, and sociology. In design, they are often applied to generative architecture.

For instance, Herr and Kvan [27] proposed the application of cellular automata to develop designs of high-density buildings, which often suffer from copy-paste approach and disregard the human need for individuality in organizing their living space. They gave an example of a high-density city block in Japan, whose alternative variants were designed with the use of a cellular automaton combined with manual design decisions.

Krawczyk [28] indicated problems of translating patterns developed by cellular automata into architectural forms, such as lack of horizontal or vertical connection between cells, lack of architectural scale, or not enough interior space and presented some solutions.

Adem and Cagdas [29] compare two approaches to applying cellular automata in architectural design: the one by Frazer [30,31] and the one by Herr [32,33]. Frazer developed a generative design model based on human-computer interaction. Rather than being attached to a real environment, it uses abstract rules and expressions, which enables the exploration of various architectural spaces. Herr was focused on early, conceptual design stages, and redefined neighborhood and cell definitions in terms of the architectural design process.

## Genetic algorithms

Genetic algorithms are inspired by biological evolution, where due to natural selection, adaptation to the environment’s requirements increases an individual’s chance for reproduction. The reproduction step is often performed not on actual solutions (phenotypes) but on their coded equivalents (genotypes). An important role is played by random mutation, as it introduces new features to the evolving population. The natural environment is replaced by the fitness function, which evaluates phenotypes. A genetic algorithm is then a strategy of exploring the genotype space, which is mapped into the phenotype space, in search of optimal solutions.

Genetic algorithms are used to solve problems of optimization as well as a generative tool. In design, they have been used among many others by Von Buelow et al. [34], who proposed a method of optimization of complex structural systems for basic parameters such as weight or number of components. In their approach, a genetic algorithm was combined with a finite element analysis and parametric modeling software. An interesting problem in design is space structures, which are frames that cover large surfaces such as supermarkets or stadiums without intermediate columns. Among others, Salajegheh et al. [35] propose a genetic algorithm that optimizes such structures with regard to their topology and geometry. Another application of genetic algorithms in design is floor plans. For instance, Carta et al. use a genetic algorithm with graph-based genotypes as a tool for the optimization of floor plans in care homes. The fitness function criteria regard the existence and accessibility of public spaces such as laundry and kitchen, as well as residents’ private bedrooms.

## L-systems

L-systems are string rewriting systems developed by the biologist Aristid Lindenmayer [36]. Similarly to shape grammars, an L-system uses a set of rules and recursion, however instead of shapes its alphabet consists of strings. The generated language also consists of strings and can be mapped into a language of shapes. L-systems were originally used to model the behavior of plant cells. Because their growth occurs simultaneously, parallelism is an essential feature of L-systems [37].

L-systems are frequently applied in modeling plants. Prusinkiewicz and Lindenmayer, in their book “The Algorithmic Beauty of Plants” [38], presented a study on computer simulation of plant development. L-systems give also interesting results when applied in architectural design. As Michael Hansmeyer points out, they are "particularly suitable to forms that contain branching, recursion, and modularity" [39]. On his website, he presents some architectural interpretations of strings generated by L-systems.

## Swarm intelligence

It can be observed that large groups of organisms exhibit intelligent behavior, even when the intelligence of the individuals is low. For instance, bees can produce honey, which is a complex process that requires cooperation and following the procedure. It can be assumed that a single bee does not know the honey recipe. Instead of this, it acts instinctively and performs simple tasks, which include interaction with other bees. Based on such behavior, a concept of swarm intelligence has been developed and applied in artificial intelligence, originally by Beni and Wang [40]. AI systems equipped with swarm intelligence use a large number of simple agents that act according to the three rules:

1. cohesion – they remain close to each other,
2. separation – they avoid collisions,
3. alignment – they move in the same direction [41,42].

As a result of the agents’ actions emergent properties of the whole system occur. This allows for finding optimal solutions to problems unsolvable for individual agents.

Among other applications, swarm intelligence is used in computer-aided architectural design. Carranza [42] presents two approaches to swarm intelligence in generating architectural forms. In the first one, a swarm is equipped with knowledge about its environment, i.e., it avoids obstacles. In the second approach, the ability of swarm learning is implemented, as the agents leave traces. Kartal and Kartal recall examples of swarm intelligence in both human and artificial art and design [43], like swarm-based projects for the architectural design of Joshua M. Taron and Tyler Julian Johnson. The Seroussi Pavilion Project, designed by Alisa Andrasek, uses self-modifying patterns of vectors based on electromagnetic fields [44]. Other examples are “The Sequence”, Arne Quinze’s giant wooden sculpture near the Flemish parliament [45], and the “Swarm Chandelier” by Zaha Hadid, in which the only source of light are hundreds of reflective black crystals suspended in petal-like volumes.

## Shape grammars

Shape grammars are systems that generate shapes with the use of recursion. They have been developed by Stiny and Gips, who were inspired by Chomsky’s string-rewriting grammar. A shape grammar is defined over an alphabet of shapes such as points, lines, planes, and solids [46], and generates a language of shapes. It consists of an alphabet of non-terminal shapes (markers), an alphabet of terminal shapes, a set of rules, and an initial shape composed of terminal and/or non-terminal shapes. The process of generation starts with the initial shape and is based on replacing the non-terminal shapes according to the rules [47], which may be applied one by one or simultaneously. A common approach is to use parametric shape grammars, i.e., grammars in which shapes have parameters that can be adjusted dependently on their context [48].

Shape grammars are often used to generate designs in a given style, as modification of rules order does not usually alter the character of obtained shapes. The Palladian grammar developed by Stiny and Mitchell [49] is one of the most famous examples. It was constructed on the basis of Villa Foscari (Villa Melancholia) in Mira, designed by Andrea Palladio, and applied for the generation of floor plans of similar villas. Many other shape grammars are used to define the character of buildings, such as Buelincks’ grammar of Christopher Wren’s City churches [50] or Downing’s and Fleming’s grammar of the bungalows of Buffalo [51].

A spectacular implementation of shape grammars for architectural design is a program CityEngine, developed by Esri company [52]. Among other applications, City Engine allows the parametric design of urban environments by means of user-defined rules. Figure 30 presents an example of a generated scenery provided by the producer.



Figure . Urban scenery generated by CityEngine [52]

For style representation, shape grammars seem the most powerful technology from the ones presented above. However, they have limitations that impede their usage. McDermott [46] points out that computer representation of shape grammars is very difficult and each implementation is tied to the particular grammar, with no universal solutions. Another important issue is difficulty in recognizing emergency. What is more, the application of shape grammar rules is associated with the task of image recognition, especially the sub-shape recognition problem. McDermott notices that this may lead to surprising results in the application of the grammar rules.   
Figure 31 a) contains a grammar rule that changes an open rectangle into the closed one.   
Figure 31 b), the rule has been applied to the sub-shape of the open rectangle instead of the whole shape. Shape grammars lack mechanisms that could prevent such events.



a)



b)  
Figure . Surprising results of applying a shape grammar rule

These restrictions may lead to the exploration of other methods of style and character representation, like graph grammar.

## Graph grammars

Graph grammars are a formalism of graph transformations. Their main component is a set of productions, each of whom can be described as (L, R, E), where L and R are graphs and E is some embedding mechanism. Whenever an isomorphic copy of L occurs in a host graph H, it can be replaced with an isomorphic copy of R. During the replacement process all the edges connecting the copy of L with the rest of the graph H are removed. The embedding mechanism E is then used to connect an inserted copy of R with the rest of H [53].

Graph grammars provide a powerful generative tool, which can be mathematically modeled with high precision. This makes them an interesting option for generative design, especially since their application allows for going beyond an initial conceptual space of design. Grabska et al. [54] proposed the application of graph grammars to generate space layouts of designs. The designs are represented by attributed graphs which provide information about both the topological structure and semantic properties of designs. The graph grammars are organized in systems and act in parallel on the design graph, which results in multiple design solutions. Another example of the application of graph grammars is provided by Du et al. [55], who proposed modeling of a product family with their use.

## Artificial neural networks

The architecture of an artificial neural network (ANN) is inspired by a brain. It consists of elements called neurons that are simplified models of biological neurons and are able to process data. The connections between them imitate synapses and are parametrized with so-called weights that are modified during the so-called learning process. As a result of processing the input signals, ANN produces the output signals. The artificial neurons are usually arranged into layers: the input layer, which collects the input signals, the output layer responsible for emitting the resultant signals, and some number of hidden layers between them which process the information [56].

The tasks realized by ANN, originally limited to classification, nowadays include both analysis and synthesis of complex problems. So-called deep neural networks, equipped with a large number of hidden layers, are developing rapidly and give results resembling the effects of human thinking. This makes them another effective tool for computer-aided design.

Although there are not many approaches to architectural design with the use of ANN [57], the recently developed systems like Midjourney [58] or Dall-e [59] contribute to this area by the possibility of generating images based on the provided keywords, which may contain references to architecture. For instance, Figure 32 presents three images generated by Midjourney in response to the description: *a temple in winter, people leaving, parametric design, nature-inspired architecture*.



Figure . Example of images generated by the Midjourney program

.

## Other approaches

The methods presented above are the most popular ones and do not exhaust the wide subject of generative tools in design. Other evolutionary approaches and other agent-based models, including curious agents [60], are just a few of them. The mentioned applications of generative methods in design are selected examples among a large number of ideas and implementations. It should be also noticed that many designers combine different methods, like Celestino Soddu in his generative art and architecture. Soddu’s software Basilica generates digital artifacts that refer to the urban medieval environment. To obtain this goal, multiple algorithms work together: “In Basilica I used specific geometric parametric algorithms, algorithms managing the transformation of event's figure by moving from a dimension to another, Cellular Automata and parallel progressions of transformations of single events that dynamically interact with others, as flocking of birds, and structures of repetition of the same algorithm applied to the same event, as fractal approach. But none of these methods is primary. The peculiarity of my approach is ‘how’ I use them all together [61].” Another software of Soddu, Argenia, is aimed at “industrial production of unique and non-repeatable objects [61]”. Figure 33 presents coffee pots generated by Argenia presented in [61].



Figure . Coffee pots generated by Celestino Soddu’s software Argenia [61]

A common approach is to use evolution-based programming to explore a space of designs created with the means of other generative methods. For instance, McDermott [46] proposed an interactive evolutionary 3D design system based on graph grammars. In his solution, the nodes of the derived graphs were labeled with Euclidean coordinates, which simplified the mapping of the graph representation into a design.

# Chapter 4. Genetic Algorithms in Creative Design

## Creative design

From the very beginning of the computer-aided design domain, there has been a need to provide a computational model of creative design, so that the human process of creation could be imitated by computers with satisfactory results. To achieve this goal, both concepts of design and creativity must be defined.

### Creativity

Computational creativity is about using a computer to generate results that would be regarded as creative if produced by humans alone [62]. Creativity is “the ability to generate ideas/artifacts that are new, surprising, and valuable” [63].

1. **New**

Boden [63] distinguishes two types of novelty: psychological (P-novelty) and historical (H-novelty). The P-novelty occurs every time a person invents, discovers, or achieves something new to them. The baby's reaching for a rattle for the first time is an example of P-novelty. H-novelty is a special case of P-novelty and takes place when a concept is regarded as new not only by its author but by other people as well. The invention of the wheel was an instance of H-novelty.

1. **Surprising**

There are three types of creativity [63], each of them arousing a sense of surprise differently.

* 1. Combinational creativity is based on combining existing concepts to generate new one. In this case, crossing over predictable elements of familiar ideas leads to unpredictable results. An example of such creativity is a collage, where fragments of graphics are used to construct a new image.
  2. Exploratory creativity occurs when a novel solution is generated with the use of existing rules and constraints. The element of surprise results from the fact that the rules and constraints have not been tested so far before. An artwork in an existing style, e.g., an Impressionist painting, may be an example here.
  3. Transformational creativity requires modification of existing restrictions so as something impossible before is now achievable. This leads to a more spectacular effect of a surprise than in the case of the two remaining types of creativity. Developing a new style in art is an instance of transformational creativity.

1. **Valuable**Although it seems relatively easy to invent novel and unexpected concepts or artifacts by random generation, this is the criterion of value that makes creativity so complex and mysterious. No matter whether the act of creation is deliberate, as in design, or accidental, as in some scientific discoveries, it is always evaluated on its usefulness.

### Design

According to J. Gero's [64] definition, the design activity can be characterized by the following properties:

1. **Goal-oriented**The goal of design is to make the world more suitable for people through the creation of artifacts. The process of design transforms design requirements, i.e., desired functions of an artifact, into its description, which can be graphical, numerical, or textual. The design description enables further processing of the design, e.g. construction or manufacturing.
2. **Decision-making**The process of design can be modeled by variables whose values are set as a result of the designer's decisions.
3. **Constrained**As design refers to the real world, the set of variables as well as the sets of their possible values are constrained.
4. **Exploration**Design requires exploring which variables are appropriate to describe design objects.
5. **Learning**Design is associated with emergence which occurs when an implicit property can be made explicit [65]. Thus, learning about new, emergent aspects of the design is necessary during the design process.
6. **Context**The context of a design emerges from many components such as the designer's state of mind, their dialogue with the developing design, previous sketches, or interaction between the designers. [66] "Artifacts mean what their contexts permit [67]."

Design activity is then a "goal-oriented, constrained, decision-making, exploration and learning activity that operates within a context that depends on the designer's perception of the context" [64].

Based on design experiences, human designers form schemas - generalized concepts that enable the design process even when the required data is incomplete. Design prototype is an important schema for creating prototypes, i.e., designs from which further designs are derived.

### Creative design

Creative design can be then defined as a process of perturbing an existing scheme to obtain a new, unexpected result. This result should be still understandable in the current or modified context [65]. The creative design model proposed by Gero uses the design prototype schema and is based on two concepts: the introduction of new variables and the introduction of a new schema.

#### New variables

Schemas are modified by adding new variables, which is sometimes combined with deleting the existing ones. A new variable may be homogenous, which means it is of the same kind as the existing one and can be integrated by the current schema, or heterogenous when the current schema is not sufficient to integrate the new variable and needs to be developed.

There is a number of processes that add new variables to the existing schema called focus design prototype, for example:

1. Combination – addition of variables from the other design to the focus design prototype;
2. Analogy – as in the combination process, it is the addition of variables from the other design schema to the focus design prototype. However, only those variables that fulfill the criteria of analogy are added. The analogy is understood as the process in which some coherent aspects of a concept are matched and transferred into another concept;
3. Mutation – a modification of the variable set in the focus design prototype by an external process. Particularly, a new variable may be added.

Some schema variables may also be removed while others are added. Among the processes that substitute variables, there are:

1. Mutation – in the case of the substitution it is the alteration of the focus design prototype variable set by replacing some variables with new, heterogenous ones;
2. Analogy – like in the addition process, in case of substitution some variables introduced by the analogy are added, but also some variables are removed from the focus design prototype.

#### New schemas

**Emergence** is a process of substitution of schemas. It occurs when some of the variables in the schema are associated with an alternate representation. The alternate representation may be then associated with other schemas.

#### Model of creative design

The model of the creative design has been proposed in [65]. New schemas are introduced in response to the extension of variable sets with heterogenous variables. After that, the current prototype design schema can be replaced with the new one and further developed during the process of design. In case when no variables are introduced and the design can be understood by the current schema the routine design occurs. The creative design is characterized by the introduction of new variables and takes place when the current schema is sufficient to interpret them as well as when a new schema is created. When no schema can be found the design is considered useless and should be rejected.

#### Creative design and evolution

It can be noticed that the process of creative design is analogous to biological evolution and its computational counterparts like genetic algorithms. The same as in evolution, creative design is all about change. The design prototype is iteratively modified until it fulfills the designer's requirements, which are equivalents of the natural environment in biology or fitness function in genetic algorithms. Some of the processes that alter the current design prototype, like combination and mutation, also resemble the evolution processes – crossover and mutation. Genetic algorithms seem then a natural choice for computational representations of creative design.

## Genetic algorithms in design

### Evolutionary design

Biological evolution is based on four main principles [68]:

1. Individuals reproduce. As a result, more descendants are produced than they can survive, as the resources of the environment are limited.
2. Descendants inherit the features of their ancestors.
3. Diversity is introduced into the population through random mutations.
4. Individuals best fitted to the environment survive and reproduce.

This phenomenon has been able to transform simple unicellular, saltwater organisms into an amazing variety of living forms, some of them extremely complex, that inhibit almost every part of Earth. The efficiency and simplicity of evolution have inspired many people to reconstruct this process in an artificial environment. Selective breeding had been known to farmers for ages when Darwin and Wallace finally explained the mechanism of evolution in the nineteenth century [4,69]. The development of computer science allowed for the introduction of the evolution of entities completely different from organic life forms – abstract data structures used for the representation of potential solutions to a given problem. Especially, evolutionary computations have been useful in computer-aided design.

Evolutionary design is evolution performed by a computer. It is an unconscious process very similar to the biological one [70] when randomness and fitness evaluation lead to the gradual progress of results. A human designer is required, however, to define the purpose of evolution. What is designed? How to present potential solutions so they are comprehensible to the computer? Which individuals will be allowed to reproduce and how will reproduction work? When should evolution stop? All these questions need to be answered in a form of an evolutionary algorithm.

Evolutionary algorithms are widely used in computer-aided design for several reasons [70]. First of all, it has been proved experimentally in plenty of design tasks that evolution is an efficient and universal problem solver. Secondly, it resembles the human design process, in which a designer combines ideas from existing concepts to create a new solution. Evolution enables the automatic improvement of designs and the exploration of different design ideas to encounter novel, useful solutions. In fact, evolutionary computation “is all about search” [68] – the search space, filled with all possible solutions, is explored to select the most optimal ones; the mechanism of evolution is used to narrow the search area.

Besides its advantages, such as simplicity, efficiency, and creative outcome, evolutionary design has some limitations. It is wasteful: there are a large number of inadequate solutions produced in each evolution process, which need to be rejected. There is also a risk of getting stuck in local optima or plateau during the search. What is more, some solutions are extremely unlikely to be found, as their features are considered useful only when fully developed (like a wheel on an axle). However, for many design purposes, evolutionary design provides satisfactory results.

There are four directions of evolutionary design [68], which are optimization, art, artificial life, and creative design. The approach presented in this dissertation is focused on creative evolutionary design with the use of a genetic algorithm, which is – among evolutionary programming, evolution strategies, and genetic programming – one of the main types of evolutionary algorithms.

### The genetic algorithm

As with other evolutionary algorithms, genetic algorithms explore a search space of potential solutions. In the beginning, a set of potential solutions is chosen randomly to form **an initial population**. A single solution in a population is called **an individual**. Each individual consists of a **phenotype**, i.e., a set of parameters representing a solution, and a **genotype**, which is its encoded version. Encoded parameters are called **genes**. As in biological evolution, each gene may occur in different variants – **alleles**. Genes are grouped into **chromosomes**. A genotype may contain one or more chromosomes. The classical genetic algorithm encodes solutions as binary strings. In such an approach, each parameter is presented as a binary number. However, the way of encoding phenotypes varies among different implementations of the genetic algorithm. Besides binary strings, any other data structure may be useful, including graphs and trees.

A very important role in evolution is played by **a fitness function** that evaluates individuals in the population. The best fitted of them will be then allowed to reproduce, analogously to the living forms able to survive in the natural environment long enough to produce offspring. Once the individuals in the population are evaluated, some of them are chosen for reproduction in the process of **selection**.

The classical method of selecting genotypes to copy them to the **mating pool** (i.e., a pool of genotypes allowed to reproduce) is *roulette wheel selection*. The probability of selection is assigned to every genotype proportionally to its fitness value. The selection is made analogously to turning the wheel, when the wider the slice (i.e., the higher probability), the greater chance it will stop on the pointer. To maintain the constant size of the population of *n* individuals, there should be *n* draws performed to choose the parents, as each pair is expected to produce two posterior genotypes. This means there may be (and usually is) multiple copies of the same genotype in the mating pool. Such a strategy may slow down the evolution when there are small differences between the fitness values of the genotypes because in such a situation their selection probability will be similar, instead of the strong preference for the best-fitted individuals.

A method of selection called *ranking selection* is deprived of this drawback. The genotypes are sorted in descending order of their fitness value and given a rank, which is their ordinal number in the list. The number of their copies in the mating pool is then determined by the function , where is a rank.

*Tournament selection* is another selection strategy in which *k* (usually 2 or 3) genotypes are randomly picked from the population. The fittest of them is copied to the mating pool. This process is repeated until the desired number of genotypes is chosen.

Once the individuals are selected for the mating pool, they are randomly split into pairs. Then, **the crossover** is performed. The genotypes of each pair are mixed to produce offspring. The classical approach to this task requires random determination of **a locus**, i.e., a position of chromosome division, for each pair of parent individuals.

The crossover operator depends on a data structure used to represent the genotype and on the goal of the genetic algorithm. Any method of mixing genotypes is allowed providing it enables the increase of the fitness value in descendant individuals. The probability that the offspring will be more adequate than its ancestors should be high enough to provide the algorithm’s efficiency.

The same requirement regards another operator of the genetic algorithm called **a mutation**. The mutation occurs with low probability (usually no more than a few percent) just after the crossover and is a slight modification of the offspring genotype, analogically as it takes place during biological evolution. Again, any method of genotype modification is allowed providing it gives satisfactory results in the evolution process. The most common approach is to replace an allele of a random gene with another allele. In the classical version of the genetic algorithm, a digit of a chromosome is altered.

As a result of crossover and mutation, a new generation of individuals is created and the cycle of fitness evaluation and reproduction starts again. Figure 34 presents the genetic algorithm's loop: In the beginning, a population of *n* individuals is generated, usually randomly. Their phenotypes are evaluated by the algorithm's fitness function and the best of them are chosen for the mating pool in the process of selection. The individuals selected for reproduction are then matched in pairs. For each pair, the crossover is performed with a random probability of mutation. As a result, descendant individuals (usually two of them) are created. The new generation of individuals replaces the parent population (it is also possible to construct the new population in another way, e.g., by leaving the best of the parent individuals). The algorithm’s cycle starts again unless the termination condition is fulfilled.

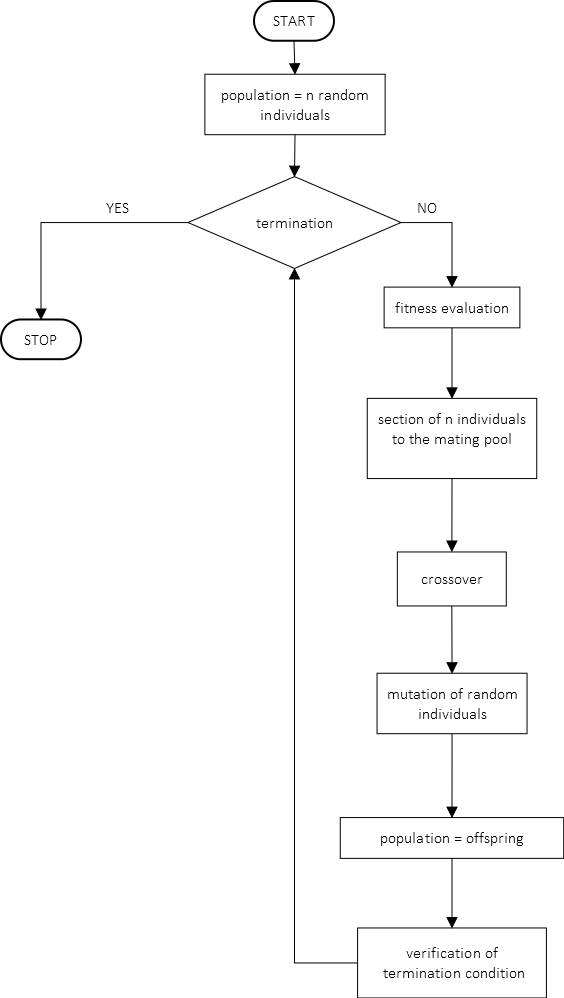


Figure . Genetic algorithm diagramm

# Chapter 5. Computational Aesthetics

“(...) a computer artist should be a programmer who can teach his computer to produce works of art by itself.”

Hiroshi Kawano, 1976 [71,72]

## Definition of computational aesthetics

"Computational Aesthetics is the research of computational methods that can make applicable aesthetic decisions in a similar fashion as humans can"[73].

The above definition was proposed by Hoenig with two restrictions. First of all, it can be noticed that aesthetics may refer to an object's content (i.e., what is communicated) or its form (how it is communicated)[73,74]. Although it is not sure whether the human brain makes such a distinction during aesthetic judgments, it seems necessary to use it for computational aesthetics. Knowledge of associations and cultural references that enable reading content is specific to people and limited to computer systems. Therefore, computational aesthetics is mainly focused on form.

Another remark stated by Hoenig limits computational aesthetics to the visual case. There have been multiple attempts to implement some imitations of human aesthetic sense in computer programs devoted to visual arts [75], including among others painting, photography, and architecture. The research devoted to the visual aspect of computational aesthetics is excessive in comparison to its other cases, for example, aesthetics of music or poetry. There are also some domains in which the application of computational aesthetics is particularly difficult, like culinary arts.

Today, after seventeen years, the restrictions of the Hoenig definition have become less obvious. More and more domains of human art and creativity have been (at least partially) translated into the language of computers. Therefore, computational aesthetics has started to play a significant role also beyond the boundaries of visual arts. What is more, it has become more often focused on the content due to the increasing possibilities of nowadays computers and algorithms.

## Measuring aesthetics

Aesthetic evaluation is an essential component of both art perception and creation [75]. Artists, in search of inspiration, evaluate the work of other artists. The process of creation consists of a sequence of decisions led by micro-evaluations. Finally, the finished piece of art is evaluated [76]. The same dependency occurs in every kind of design where aesthetics is taken into account. Evaluation seems then an important function of computational aesthetics, no matter if it regards synthesis or analysis.

### Birkhoff aesthetic measure

Although aesthetic principles in art and design have been discussed for ages, the first known attempt to formally define aesthetics was made in 1933 by Birkhoff[1]. He presented a formula of the aesthetic measure , where is “a realization that the object is characterized by a certain harmony, symmetry or order”, means the object's complexity, proportional to “a preliminary effort of attention, which is necessary for the act of perception”, and finally is defined as “the feeling of value or aesthetic measure, which rewards this effort”. According to Birkhoff, the ways of measuring order and complexity differ among domains. He specified these values for four domains: polygons, poetry, music, and Chinese vases.

For polygons, the order is defined as a composition of the relations within the figure, like vertical symmetry, equilibrium, rotational symmetry, the relation to the horizontal-vertical network, and the absence of the unsatisfactory form, when the polygon has some features perceived as not aesthetic, for example, is almost, but not fully, symmetrical. Complexity refers to the number of lines containing the polygon’s edges.

In his attempt to measure the musical quality of a poem, Birkhoff defined the order by the following factors: alliteration and assonance, which can be aesthetic or unaesthetic, depending on the case, rhyme, the presence of musical vowels, i.e., “*a* as in *a*rt, *u* as in t*u*nef*u*l, bea*u*ty, and *o* as in *o*de” and a number of consonants, which, if excessive, is perceived negatively. The complexity is the total number of elementary sounds.

The Chinese vases can be described by characteristic points on their vertical cross-sections, i.e., the terminal and inflex points as well as the points with a vertical tangent and the points of steep change of tangent direction. The order is defined by relations between the characteristic points and between the tangents, whereas complexity is the number of characteristic points.

In music, the order regards relations between notes and sequences of notes. Tonal start and close, cadence, different kinds of repetition, melodic and harmonic relations between notes, inversion, and transposition within a phrase are all components of the musical order. The complexity is defined as the total number of notes.

The formulation of the aesthetic measure by Birkhoff is regarded as the beginning of computational aesthetics. The empirical studies have shown its multiple shortages, for instance, Friedenberg and Bertamini found that among octagonal polygons, those with more concavities are considered more attractive [77]. This contradicts Birkhoff’s measure according to which the most attractive polygon is a square.

Figure 35 presents six polygons from a set given by Birkhoff as an example to illustrate his approach.

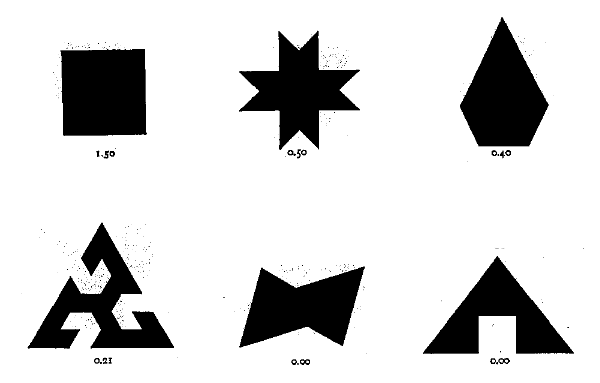


Figure . Examples of polygons evaluated by Birkhoff aesthetic measure (source: [9])

Despite its controversiality, the Birkhoff measure is still referred to and discussed, with many attempts of improvement and extension (see [6,78–80]). Among many applications, the one proposed by Megahedand Gabr [81] is particularly interesting regarding the subject of this dissertation, as it uses the modified formula for polygons to evaluate buildings. Other examples of applying Birkhoff measures in architectural design can be found in [8,10].

The contemporary research can lead to the conclusion that the main assumption of the Birkhoff measure, i.e., defining aesthetics as a relation of order and complexity, is still up to date, although the definitions of these components are hard to determine. It is worth noticing that there have been several other attempts to define aesthetics by means of information and complexity, among others the measures proposed by Machado and Cardoso, Bense, Moles, Schmidhuber, Gell-Mann and Lloyd, as well as the Audio Oracle algorithm for evaluating music designed by Dubnov and Assayag [82].

### Zipf’s law

Zipf’s law is an observation that in many types of data, the frequency of events is inversely proportional to their statistical rank. It can be formulated as , where is the quantity of occurrences of an event within a given phenomenon, is its position in a list sorted in descending order by frequency, and is close to . This means that the most frequent event will occur approximately twice as many times as the second most frequent one, et cetera. The law was popularized by Zipf [83] and originally referred to the number of words in the natural language corpus, however, it has turned out it is applicable in many other systems. For instance, Zipf noticed that the same formula refers to the number of towns’ citizens.

Although the reasons for the Zipf’s law accuracy are still not fully understood, its universality allows to assume that human aesthetic preferences are driven by the same rule. Manaris [84,85] proposed an evolutionary system for music generation. The fitness function was designed based on an analysis of over 200 sample tracks of different genres. Zipf-Mandelbrot distribution – a modification of Zipf’s law – proved to characterize aesthetically pleasing music. Chris Emmery [86] conducted an experiment in which participants evaluated simple visual art generated automatically with regard to either Zipf’s law, the golden ratio or randomly. The results have shown a preference for Zipfian distribution in art.

### Machine learning approach

Nowadays the interest in computational aesthetics is growing rapidly and new methods of measuring aesthetics are being developed among many domains. For instance, aesthetic evaluation is used by algorithms for the prediction of aesthetic ratings of photographs [75]. The prediction of ratings is helpful in cameras, it also may be applied to select, for different purposes, high-quality photographs from the internet. The evaluation is performed based on so-called features, i.e., measurable properties of a photograph. The features can be local and regard single pixels, for example, color, hue or saturation, or global and refer to the whole image, as composition or adherence to the rule of thirds. Another division is based on the level of abstraction. The low-level features as color and edges are opposite to the high-level ones, which describe the content. The classifiers like Bayes classifiers [75,87,88], decision trees, support vector machines [75,87,89], or, more recently, convolutional neural networks [75,90,91] are trained to predict human preference on the basis of large sets of photographs described by features. Tools designed primarily for computer vision are often adapted for computational aesthetic evaluation. A common approach is computing SIFT descriptors [92] to represent an image [75,91].

A comprehensive study of the application of computational aesthetics for art analysis is presented in [75]. The described algorithms are helpful in content analysis [75,93], forgery detection [75,94], detection of an author [75,95], a style [75,96], as well as of a painter’s influence [93,97], in estimating a period in artist’s life when the painting was made [75,98] and in quality assessment [75,99]. The accuracy of the algorithms’ results often exceeds the work of professional critiques, which is a great achievement of artificial intelligence. As well as in the case of photographs, the algorithms designed for painting analysis are based on classifiers that use sets of features. The low-level features describing formal aspects of paintings, such as color and edge histograms, are the most popular choice. However, there have been successful attempts to introduce the features regarding the semantics of an image[75,100,101].

## Aesthetics of architecture

As mentioned, there are two kinds of aesthetic experience: one is a response to an object’s form resulting from a human innate sense of beauty, and another is a pleasure obtained by realizing the cultural context of an object and decoding its meaning. The second, content-oriented aspect of aesthetics is beyond the scope of this dissertation. Moreover, the approach to computational aesthetics of architecture presented further will be fully focused on form.

The aesthetic outlines of architectural design are as old as architecture itself. Some of them, like proportions, are easy to compute and implement in computer programs. Others, like rhythm or harmony, refer to impressions and not only lack formal definition but also their perception varies among people and times. This makes them difficult to translate into the language of machines.

Let us recall some basic principles of architectural design.

### Proportion ratios

The golden ratio is a proportion of two quantities such that [102]. Of its frequent occurrence in nature [103] it is often assumed the golden ratio is the most pleasing proportion for humans. Even though contemporary research questions the aesthetic preference for the golden ratio (the problem discussed among others in [104]), it has been widely used in art, especially in architecture [105], for millennia. An example may be the modernist architecture of Le Corbusier, like Villa Savoye (Figure 36), designed with regard to this proportion.



Figure . Villa Savoye designed by Le Corbusier [106]

The silver ratio is defined as a proportion of two quantities such that . It is less popular than the golden ratio, however still widely used in art and design. Liu and Zhang [104] mention a classic iPad, whose length and width are in the silver ratio, as well as the length and width of the YouTube logo. In Japanese architecture, the silver ratio is more important than the golden one. Many famous Japanese buildings apply this proportion excessively, for example, Hōryū-ji temple (Figure 37).



Figure . Hōryū-ji temple in Ikaruga [107]

### Fractal dimension

“My personal feeling is that the definition of a ‘fractal’ should be regarded in the same way as a biologist regards the definition of ‘life’.”

Kenneth Falconer [108]

“Roughly speaking, a fractal set is a set that is more ‘irregular’ than the sets considered in classical geometry. No matter how much the set is magnified, smaller and smaller irregularities become visible.”

Gerald Edgar [109]

In 1975 Mandelbrot proposed the term “fractal” for a phenomenon observed for ages in nature, art and mathematics. Although everyone is familiar with fractal structures, they are hard to define. Instead of giving a formal definition, Falconer [108] described a fractal by the following properties:

1. has details on small scales,
2. both global and local structures of are too irregular to be described using traditional geometry,
3. has often some form of self-similarity,
4. the “fractal dimension” of is usually greater than its topological dimension,
5. is usually defined in a simple way, for instance recursively.

The term “fractal dimension” is used to describe the complexity of a fractal and, unlike topological dimensions, may have a non-integer value. Fractal dimension reflects how much a figure’s size changes depending on a scale – in the example given by Mandelbrot, the length of a coast increases with decreasing the length of a measuring stick.

There is a strong connection between fractals and human aesthetic sense. Spehar [110] proved that “humans display a consistent aesthetic preference across fractal images, regardless of whether these images are generated by nature’s processes, by mathematics, or by the human hand”. The research also showed the most positive response to fractal dimension between 1.3 and 1.5.

Many examples of human activity show a preference for fractals. Probably the most important one is the admiration of nature, with plants, landscapes and even animals having a fractal structure [105]. African architecture is known for its usage of fractal forms [111,112]. Figure 38 contains a plan and photography of Mokoulek in Cameroon. The pictures are presented online in [111], where one also finds a simulation of generating a fractal plan of the village. Beside physical art and design, there is also a branch of digital art based on fractals – fractal art, where images are generated by algorithms that calculate their fractal pattern.

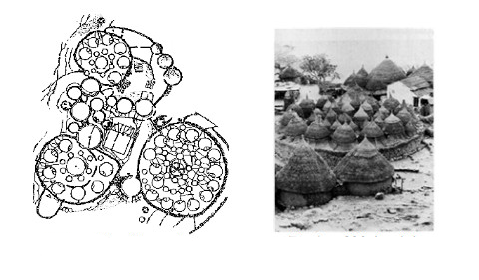


Figure . Mokoulek – a plane and a photograph of the village [111]

### Gestalt perception laws

At the beginning of the 20th century, Max Wertheimer gave rise to the Berlin school of Gestalt psychology [113]. The laws of visual perception formulated by Gestalt regard grouping and figure-ground organization. Contemporary psychology verified these ideas experimentally, which resulted in assigning priorities to the laws and relating them to other visual processes like shape and depth perception, as well as adding new principles [114].

The Gestalt laws of grouping list circumstances in which figures are perceived as a group. The examples recalled in [114] are proximity (when figures are close to each other), similarity (when figures share the same feature, like size or color), common fate (when they move in the same direction), symmetry, parallelism, closure (when figures form a closed shape), common region (when they are inside the same shape), and continuity (or good continuation, when figures contain lines that are a prolongation of each other).

The figure-ground organization principles provide properties of a figure to be discriminated from its background. They are convexity, symmetry, small area, and surroundedness.

Gestalt psychology influenced Bauhaus, an art school founded in Weimar by Walter Gropius [115]. Bauhaus student, Hans Beckmann, noticed [116]: “Realizing that art, as such, cannot be taught, the emphasis may be on visual education, partly based on Gestalt psychological insight, with an aim to achieve what Josef Albers calls: ‘A meaningful approach to the production of the form [117,118].’”

The direct influence of Gestalt can be seen in Polish architecture after the Second World War when Juliusz Żórawski published his considerations regarding the application of psychological laws in architectural design. According to Żórawski, the architectural form should respond to basic human pursuits, such as forming (i.e., seeing form in everything), geometrization, finished numbers, and strong and compact form. Therefore, the most appropriate form is simple and unequivocal. This can be achieved by outlining the main point, short, symmetrical, and steady rhythm, horizontal or vertical orientation, bottom-top direction, outlining the bottom and the top of the composition, clarity of the form standing out on its background and outlining the form’s culmination. Żórawski also mentioned the value of asymmetry and irregularity, which, although contradicting the main human preferences, can also bring a sense of beauty [116,119].



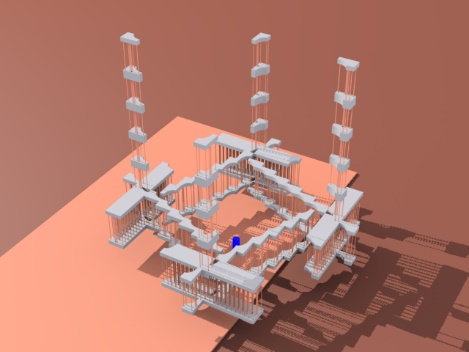
Figure . *Szklany Dom* in Warsaw designed by Juliusz Żórawski [120]

### Aesthetical components of the architecture

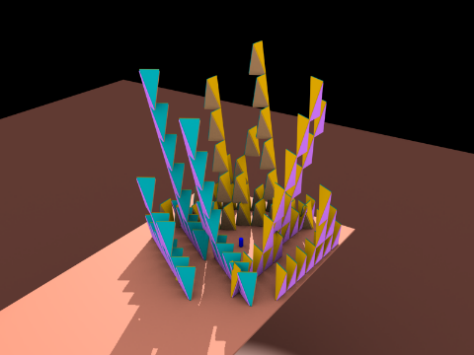
Although styles and directions have changed multiple times throughout centuries, there can be indicated some aesthetic qualities [121,122] that architectural forms regarded as beautiful have in common, no matter if we speak about ancient Greek theatres, gothic churches, or modernist villas. These are among others order, chaos, coherence, complexity, balance, clarity, and unity. Order is essential to invoke a feeling of comprehension, however, it reduces information and in the extreme leads to maximal entropy. To prevent this and keep an object meaningful, some doze of chaos is necessary. Coherence is the quality of components belonging together, which can be obtained by different kinds of ordering or by sharing common features. Unity is a concept related to coherence, however, its focus is rather on maintaining a coherent structure while providing enough ambiguity to keep a recipient interested. Complexity, “a direct metric for informational content” [121], can be perceived positively or negatively, depending on the amount of provided information and its organization. Balance is connected to a human innate sense of stability and results from equal distribution of visual weight. Finally, clarity is a characteristic of being distinguishable.

All these abstract qualities are obtained by manipulating aesthetic components, like contrast, which is a result of arranging elements with opposite features, for instance, size (big-small), color (black-white), or orientation (horizontal-vertical). Another example is proportion, a relationship between two things having different sizes. Various proportions have been used in architecture, like the mentioned golden and silver ratios or the Modulor – a scale of proportions based on the human body, developed by Le Corbusier. One of the most important components is symmetry, strictly associated with the sense of beauty, which is “an operator acting on an object, where the defining feature is that the object remains unaltered” [123]. An object is symmetrical when it is invariant under some transformations, like rotation or reflection. Rhythm, obtained by repeating elements, allows perceiving an architectural form analogically to a piece of music. A concept related to rhythm is a pattern, a repeating shape or form that fulfills a surface.

Figure 40 contains examples of architectural forms generated by Gardner & Krishnamurti with their software fs [121] with regard to aesthetic criteria.



a)



b)

Figure . Architectural forms based on symmetry, generated by Gardner & Krishnamurti’s software (screenshots published with author’s permission)

# Chapter 6. Application of Computational Aesthetics in Generative Architecture Design

In many cases of automated design, generative systems ought to be equipped with some functionality that enables the software to make decisions corresponding to human aesthetic preferences. The main difficulty here is to provide a computational definition of aesthetics. Differences in aesthetic sense among cultures and societies as well as individual people, or even the same person in different periods of their life, make this task even more challenging. The controversies around Birkhoff's formula mentioned in the previous chapter prove that the discovery of the universal aesthetic measure is, at least at the moment, unreachable. However, computational aesthetics is still widely used for many purposes, among others generating designs in a given aesthetic style or character. The presented solution, focused on architecture design, uses a two-way approach to accomplish this goal. At first, the reference building provided by the designer is analyzed and on its basis, a graph grammar is constructed. The graphs resulting from the application of the grammar rules represent architectural forms that share some aesthetic features with the reference building, like the components they are made from. Secondly, the obtained buildings are evolved by the genetic algorithm. Its fitness function evaluates the overall aesthetics of design and is based on Birkhoff's approach. However, instead of searching for the holy grail of the universal aesthetic measure, the author has focused on identifying aesthetic constituents and providing a method for comparison of architectural forms. This means that no discussion is raised over the right proportion between complexity and simplicity, chaos and order, etc. Instead, the fitness function verifies if the level of an aesthetic factor is similar in the compared objects.

Aesthetic coherence between the reference building and the derived forms is then ensured not only syntactically by a graph grammar but also on a more abstract level in the process of evolution.

## Aesthetics in generation

Methods of constructing graph grammars based on graphs that represent architectural forms will be discussed in Chapter 7. The Genetic Algorithm for Generative Design of Architectural Objects in a Given Aesthetic Character. The graphs resulting from applying the graph grammar rules represent derivative buildings. Let us list the aesthetic qualities that architectural objects represented by graphs obtained this way share with the reference building.

### Component types

As the grammar rules reflect the steps of designing the reference building, they consist of the same components. Of course, the random application of the rules results in a different configuration of these components. Figure 41 shows a reference building composed of two types of components: a cube and a cuboid. Figure 42 contains buildings generated on its basis.

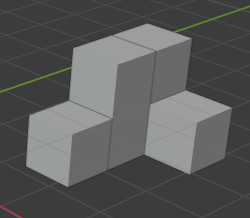


Figure . A building made from two types of components

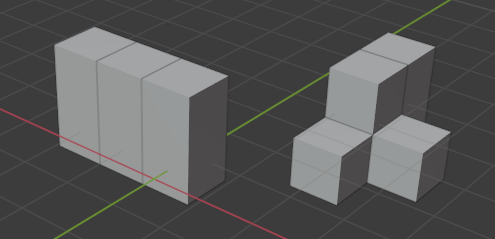


Figure . Buildings generated by the grammar obtained from the building in Figure 41

To introduce more variety into the derivative forms, some modifications of the components are allowed.

#### Modification of component instances

A graph generated by a graph grammar consists of nodes with two groups of attributes assigned. The first one, non-accidental attributes, defines the corresponding component type. The second one, metric attributes, defines an instance of the given type. The presented generative algorithm randomly modifies the metric attributes assigned to nodes of generated graphs. As result, the aesthetic character of the object defined in a large degree by component types is preserved while some novelty is added. The buildings from Figure 42, in which the metric properties have been modified, are shown in Figure 43.

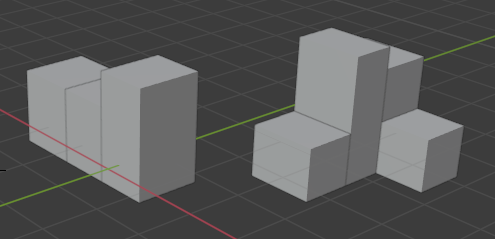


Figure . Modification of metric properties of the buildings from Figure 42

#### Modification of component types

The total change of component types does not preserve the aesthetic character of the building, as shown in Figure 44. The components of the building from Figure 42 have been replaced with random ones, so that only the relations between the components remain the same. However, not every modification of component types results in a rapid change of aesthetic character. The component type is designated by a set of non-accidental properties, divided into two subsets – the more essential main properties, and the additional detailed properties. It is then possible to easily define a grade of difference between two component types.

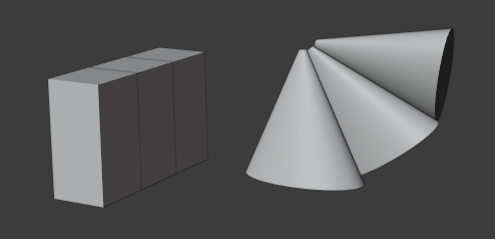


Figure . Buildings with different component types ()

A **grade of difference** between two component types is defined as , where is the number of main non-accidental properties whose values differ for these types, and is the number of detailed non-accidental properties whose values differ for them.

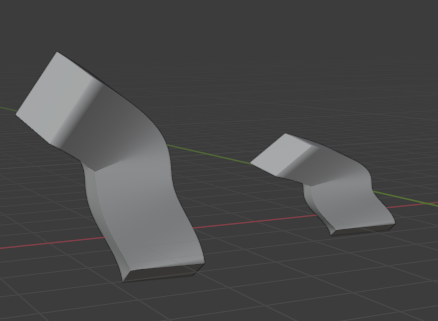


Figure . Example of

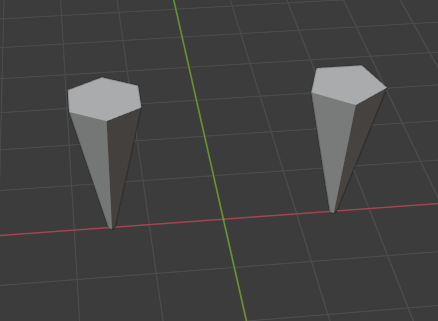


Figure . Example of

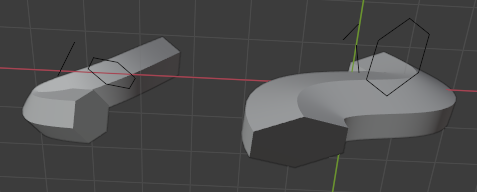


Figure . Example of

The above figures contain instances of some component types. Because the types of the components in Figure 45 are the same, their . The components in Figure 46 differ with one detailed non-accidental property, the number of edges. Therefore, in this case, . Figure 47 contains an example of , as the components differ with one main non-accidental property, the cross-section symmetry, and one detailed property, the axis curvature.

If the grade of difference between the corresponding components is low enough, it is possible to maintain a similar aesthetic character of two architectural forms. The grade of difference of the component types in buildings from Figure 44 is equal . Figure 48 presents two buildings in which this parameter is equal to . The difference between the aesthetic characters of these buildings is not as great as of the ones in Figure 44.

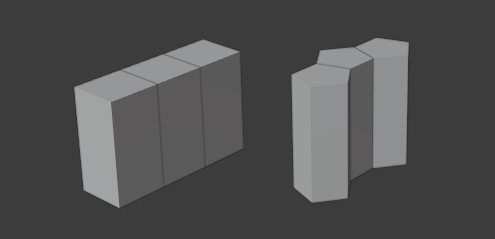


Figure . Buildings with different component types ()

This observation has led to the decision of equipping the generative algorithm with the optional functionality of creating additional grammar rules, in which some non-accidental attributes of nodes representing components are altered. The allowed grade of difference between the component types is parametrized.

### Relations between components

The rules of the generated graph contain the same relations between components as the reference building they are based on. It should be noticed that the end-to-end relation may occur only between the components having the basis of the same shape. Therefore, in case of creating an additional rule by altering some non-accidental attributes of nodes, as described in the previous section, the validity of such a rule should be checked. Figure 49 presents an instance of a rule that attaches a component in an invalid end-to-end relation. This rule would be rejected by the system.

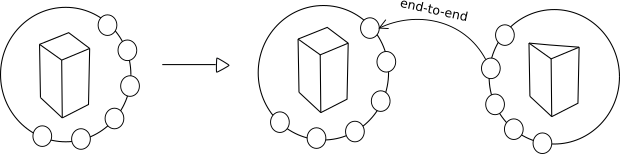


Figure . Invalid production rule

### Analogical rules

When a graph grammar is constructed based on the reference building representation, its rules reflect the designer’s steps of attaching components. For instance, a simple building from Figure 50 emerged from connecting a cube with a cuboid. A grammar rule that represents this action is shown in Figure 51 – a bond representing the cube surface is in the end-to-end relation with a node that represents the surface of the cuboid. It can be noticed that this rule enables attaching the cube to only one of the four identical surfaces of the cuboid. This limits the number of possible derivative solutions without significant impact on their aesthetical character. Therefore, the default functionality of the generative system provides the creation of so-called analogical rules, which include representations of all the identical surfaces of the basic component, provided they are all the component’s side surfaces or all the component’s bases. Figure 45 presents an extension of the grammar from Figure 51 with three analogical rules, which enable attaching the cube to the remaining side surfaces of the cuboid. Figure 53 contains a building obtained this way, which was impossible based only on the rule from Figure 51.

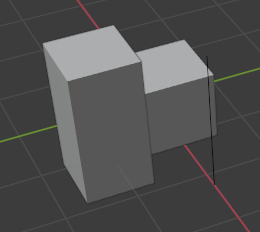


Figure . Building made from two components

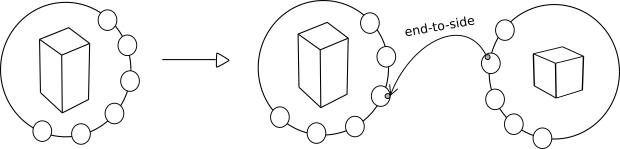


Figure . Grammar rule attaching components from Figure 50

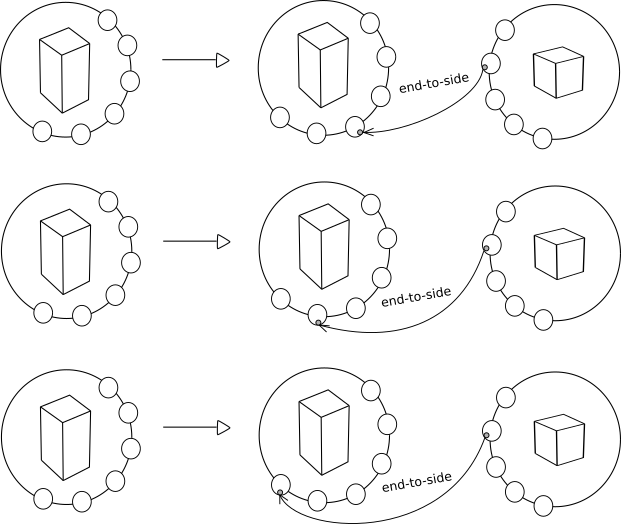


Figure . Analogical rules based on the rule from Figure 51

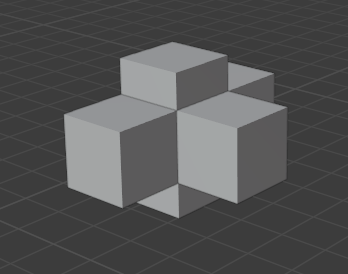


Figure . Building obtained with the use of analogical rules

### Reverted rules

Another (optional) functionality of the presented generative tool is the construction of so-called reverted rules. This means that when a rule of attaching component to component is created, it is automatically followed by the creation of a rule that attaches to . This option enables to create more complex forms. Figure 54 shows a reverted rule based on the rule from Figure 51. Adding this rule to the graph grammar based on the reference building shown in Figure 50 enables the creation of buildings such as the ones in Figure 55. In this case, the usage of the reverted rule introduces more diversity into generated solutions while preserving their aesthetical character.

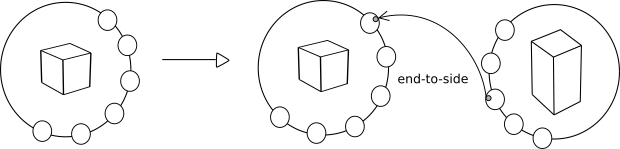


Figure . Reverted rule based on the rule from Figure 51

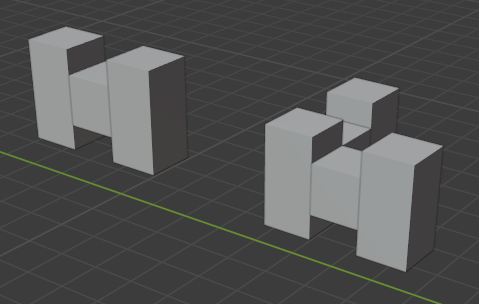


Figure . Buildings obtained with the use of a reverted rule

### Patterns

In many cases, single components do not represent the aesthetic character of a building to a satisfactory degree. Instead, the characteristic parts of an architectural object, like a balcony or a tower, may be a configuration of several components. Grouping components of the reference building into patterns results in regarding them in the generated graph grammar. Instead of a number of rules that attach the components of the pattern, only one rule of attaching the whole configuration is created. Figure 56 shows an example of a reference building with a pattern of a tower marked in red. Figure 57 contains a grammar rule constructed on the basis of this example, and Figure 58 some derivative objects. For comparison, Figure 59 presents buildings generated with a grammar that does not regard the tower pattern. In this second case, the aesthetic character of the reference building has been lost.

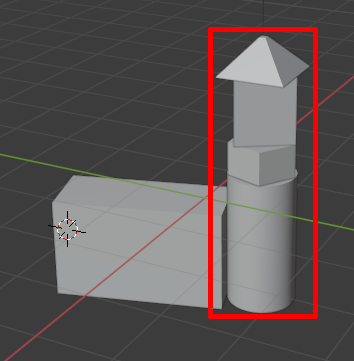


Figure . A reference building with a pattern

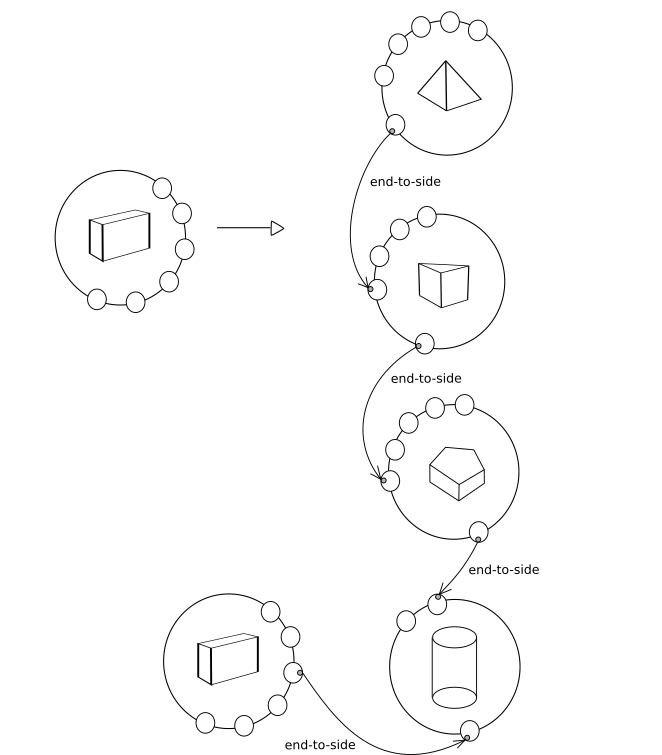


Figure . A grammar rule of attaching a pattern

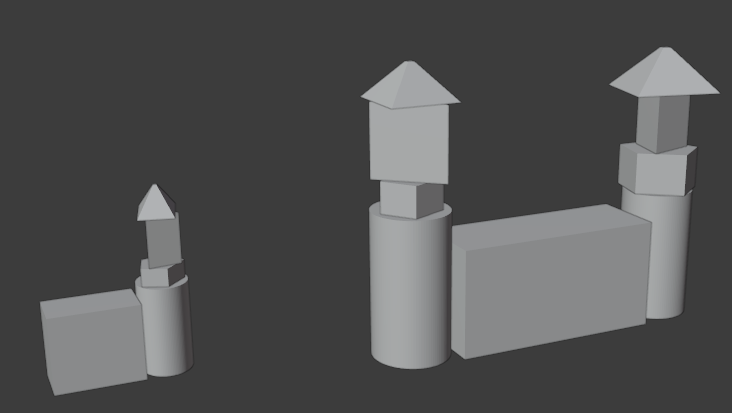


Figure . Buildings generated with the use of the rule from Figure 57

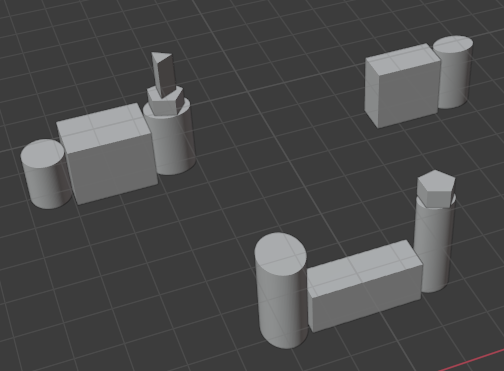


Figure . Buildings generated with the use of the grammar without patterns

### Number of components

The number of components is another important aesthetic feature. Figure 60 shows a reference building composed of seven components and Figure 61 some derivative buildings, each of them made from several components. Figure 62 contains objects obtained with the use of the same grammar, but having a much larger number of components. It can be seen that their aesthetic character is different.

In the presented generative system, the number of components in the derivative buildings is parametrized.

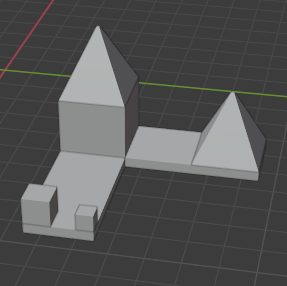


Figure . A reference building made from seven components

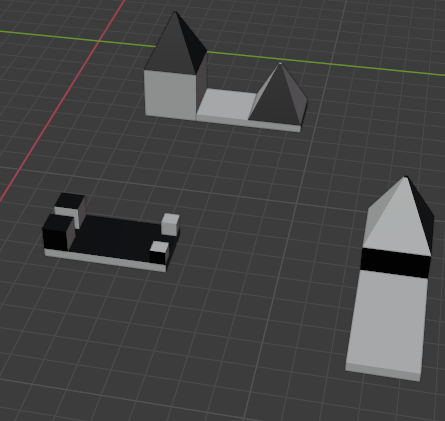


Figure . Derivative buildings made from several components

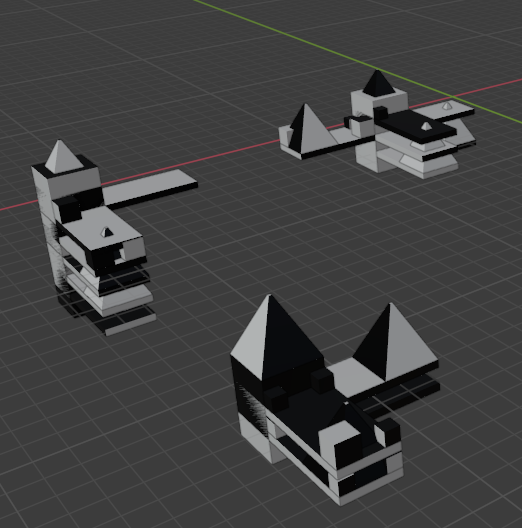


Figure . Derivative buildings made from many of components

### Multiple reference buildings

In most cases of architectural design, context is one of the most important factors an architect needs to take into account. Including references to other designs gives interesting and meaningful results. The described approach allows defining the aesthetic character of more than one reference building. The created grammar contains rules that construct all the reference buildings. In consequence, the derivative objects contain a mix of aesthetic features of the provided examples.

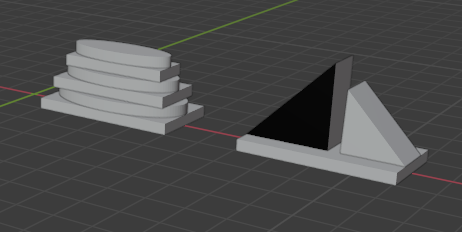


Figure . Two architectural forms used as reference buildings

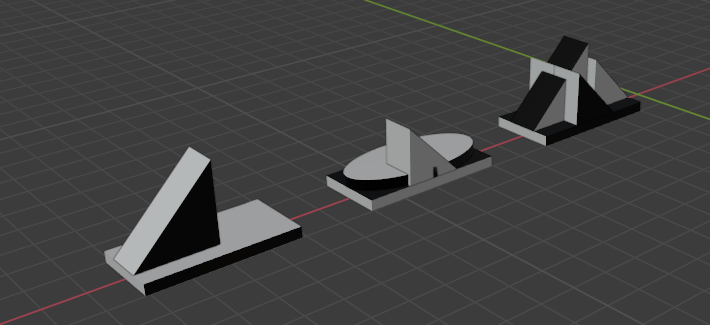


Figure . Objects generated on the basis of two reference buildings

Figure 63 shows an example of two reference buildings, and Figure 64 some derivative forms.

## Evaluation of aesthetics

The genetic algorithm of the presented generative system uses a fitness function focused on the aesthetic properties of the evaluated object. The aesthetic value of an object is calculated based on order and complexity elements, and compared to the aesthetic value of the reference building. The following aesthetic factors are considered:

1. The elements of order:
   1. alignment (arrangement of components into comprehensible rows and columns, or diagonal lines),
   2. mirror symmetry (the type of symmetry detected by a human brain the most readily [124]),
   3. rotational symmetry (another symmetry that plays a significant role in architecture),
   4. stability (the ability not to fall down);
2. The elements of complexity:
   1. number of components,
   2. number of component types.

The idea of the proposed evolution is to obtain derivative buildings with a similar aesthetic value as the reference building.

### Alignment

The level of alignment of the analyzed building (or buildings) is measured based on planes determined by the building faces. The number of alignment relations is calculated as described in Appendix A: The algorithms for determining the number of order relations. The level of alignment is then computed according to the formula , where:

1. is a set of alignment relations, where each relation is defined by a number of at least two faces aligned to a given side of a given plane,
2. , where and is the number of faces aligned to the plane and is the total number of components in the evaluated building,
3. is an alignment factor.

Using enables rewarding buildings in which multiple faces are aligned to the same plane, which gives the observer a greater sense of order and a lower sense of complexity than in a situation when fewer faces are aligned to multiple planes. Figure 65 and Figure 66 illustrate these cases. The building in Figure 65 contains components arranged in two rows, while in Figure 66 – in one row. The alignment value for each building has been computed as and , respectively, with the alignment factor set to .



Figure . Building components arranged in two rows

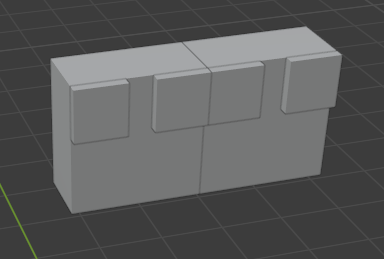


Figure . Building components arranged in one row

### Mirror symmetry

To compute the level of the mirror symmetry in the analyzed building one needs determine the number of partial mirror symmetries, i.e., to find all the planes of partial symmetries and count the components assigned to each of them. This task is realised by the algorithm presented in Appendix A: The algorithms for determining the number of order relations.

Having a set of all the partial mirror symmetries containing at least two components, the total mirror symmetry value can now be computed analogically to the alignment value, with the use of the formula: , where:

1. is a set of partial mirror symmetries where each relation is defined by a number of at least two components symmetrical against a given plane,
2. , where and is the number of components symmetrical against the plane and is the total number of components in the evaluated building,
3. is a mirror symmetry factor.

Again, with a large number of symmetry planes increases the object’s level of complexity and decreases its level of order.

Figure 67 presents a building in which partial symmetry can be detected. The symmetry value computed by the software is , with the mirror symmetry factor set to .

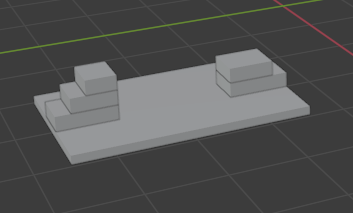


Figure . Partial mirror symmetry of a building

### Rotational symmetry

The level of rotational symmetry is computed analogically to the level of mirror symmetry - the axes of symmetry are collected along with the symmetrically arranged components. The algorithm for finding partial rotational symmetries among a group of components is based on Brass [125] algorithm for finding maximum-cardinality symmetric subsets of a set of points and is presented in Appendix A: The algorithms for determining the number of order relations.

Again, the total symmetry value is computed with the use of a set of partial symmetries containing more than one component. The formula of the rotational symmetry of a building is , where:

1. a set of partial rotational symmetries where each relation is defined by a number of at least two components symmetrical around a given axis,
2. , where and is the number of components symmetrical around the axis and is the total number of components in the evaluated building
3. is a symmetry factor.

In the building shown in Figure 68 all the components take part in the rotational symmetry around the central axis. With the rotational symmetry factor set to , the rotational symmetry value for this building is .

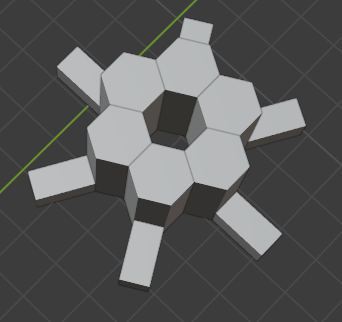


Figure . Rotational symmetry of a building

### Stability

The stability of an architectural object is essential not only for obvious functional reasons but also for the aesthetics of an object. In Birkhoff’s aesthetic measure for polygons, the rating depends on the figure’s being in equilibrium. The presented system disqualifies forms that seem unable to stand on the ground without falling, as the one shown in Figure 69. The equilibrium is checked by the following algorithm:

1. A center of mass of the evaluated building is calculated based on its components’ vertices .
2. By traversing , a subset of points is found such that , where and denote the coefficient of and , respectively.
3. By determining maximal and minimal values coefficients and among , a rectangle circumscribing the building’s base is constructed.
4. The projection of point is onto the plane containing is created. If , the building is considered stable. Otherwise, the building is unstable.

The value of stability is for stable objects and for unstable ones. Figure 69 and Figure 70 present buildings with and , respectively.

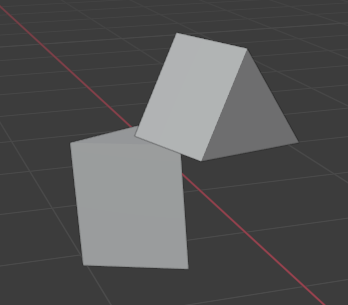


Figure . A building lacking stability

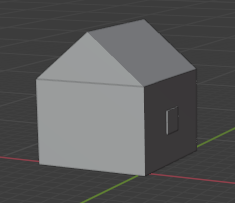


Figure . A building in equilibrium

### Number of components

The component’s number is probably the most obvious aesthetic factor that influence the observer’s sense of complexity. This value is defined during the process of generation, as described in 1.17.6. However, since some grammar rules of the generated building may not be possible to apply, which results in a not sufficient number of components, further verification seems necessary. Therefore components number is one of the factors processed during evaluation.

Figure 71 shows a building with , while in Figure 72 .

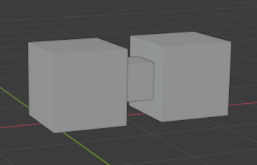


Figure . A building made from three components

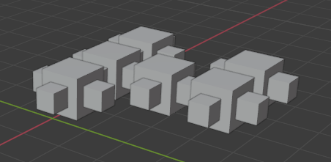


Figure . A building made from thirty components

### Number of component types

Another indicator of the object’s complexity is the number of different component types . Both buildings in Figure 73 and Figure 74 are made from the same number of components. The first building seems more complex due to the fact it contains three types of components (), while all the components of the second building are of the same type ().

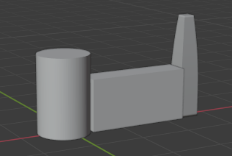


Figure . A building containing three types of components

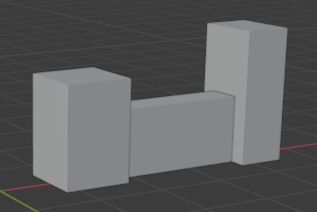


Figure . A building containing one type of components

### Aesthetic measure for comparison of architectural forms

The evaluation of a derivative form is performed by calculating how much aesthetic factors of the evaluated building differ from the ones of the reference buildings.

For each of the aesthetic factors: the alignment, the mirror symmetry, the rotational symmetry, the number of components and the number of component types – its average value in the set of the reference buildings is computed. For instance, if a set of the reference buildings contains two objects, one built from three components and another one built from five components, the arithmetic average for the number of components factor in this set is .

Let a sequence contain arithmetic averages of the individual aesthetic factors of the reference buildings: the alignment, the mirror symmetry, the rotational symmetry, the number of components and the number of component types, respectively. Let contain the values of the same aesthetic factors of the evaluated building. The aesthetic measure of the evaluated building is computed according to the formula: , where:

1. is a stability value of the evaluated building,
2. for each and , where {1,…,5}

Figure 75 presents a reference building, while Figure 76 and Figure 77 show some derivative forms. For Figure 76 the value is equal to . In Figure 77 .

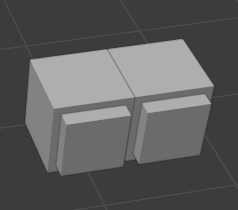


Figure . Reference building

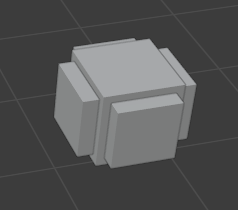


Figure . Building derived from the building in Figure 75 ()

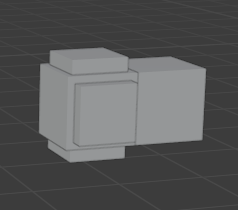


Figure . Building derived from the building in Figure 75 ()

# Chapter 7. The Genetic Algorithm for Generative Design of Architectural Objects in a Given Aesthetic Character

Having introduced all the necessary tools – the model of visual perception, the composition graphs along with the graph grammars, the aesthetic components of design and the methods of aesthetics evaluation – one can now present the genetic algorithm that uses all these elements to generate architectural forms in a given aesthetic character.

The structure and functioning of the genetic algorithm have been described in subsection 1.13.2. Further sections of this chapter contain the implementation of its components.

The genetic algorithm has been implemented in Python as a Blender add-on. Its main functions are included in Appendix B: The source code of the genetic algorithm.

## The individual

Each individual in the presented algorithm is a design of a single building. Such a design consists of its visualization, a phenotype, and its internal representation, a genotype. The population of individuals evolves to satisfy the design criteria and gain the aesthetic character of the reference buildings.

## The phenotype

The phenotype is a configuration of basic components, specified in subsection 1.1.2 along with the possible relations between them.

Each component is characterized by properties. The main non-accidental properties, i.e., the cross-section edge, the cross-section symmetry, the cross-section size and the axis type, define the component type. The detailed non-accidental properties, e.g., the number of cross-section edges, define its subtype, while the metric properties, like width or height, refer to the component instance.

The relations between components regard connections of components’ surfaces and can be end-to-end (when one component prolongs another, i.e., they share a common surface) or end-to-side, when one component is attached to another one. The end-to-side relation is further divided into subtypes like the center-to-center, the edge-to-edge and the irregular one.

Figure 78 presents a phenotype of a building made from two components in an end-to-end relation. The type of bottom component is characterized by a polygonal cross-section edge, the cross-section invariant under reflection, constant cross-section size, and a straight axis. The top component type has a polygonal cross-section edge, the cross-section invariant under reflection, variant cross-section size and a straight axis.

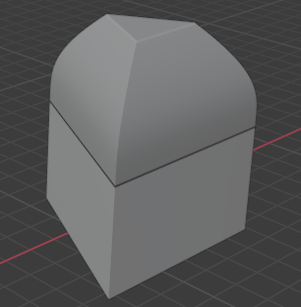


Figure . Phenotype of a building containing two components

## The initial population

### The graph representation of the reference buildings

The initial population for the genetic algorithm is created based on provided examples, i.e., reference buildings.

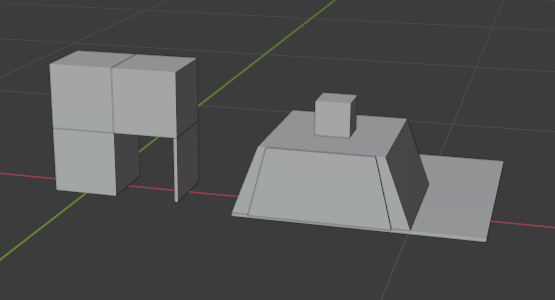
A reference building is created by the designer as a configuration of basic components which can be described analogically as in the case of the phenotype. Once a reference building is provided, the software performs its analysis to construct the internal representation in a form of a CP-graph (the CP-graph definition is presented in 1.2). The graph nodes represent components and are attributed by a set of properties characterizing each component. The node bonds represent component surfaces and are attributed with surface areas, like the list of edges, the central point coordinates or the coordinates of other points essential for connections between components. The edges refer to the relations between components and connect two bonds representing surfaces. In the case of the end-to-side relation, the bond attributes are used as the relation’s arguments. Each building is represented by one graph.

Figure . Reference buildings

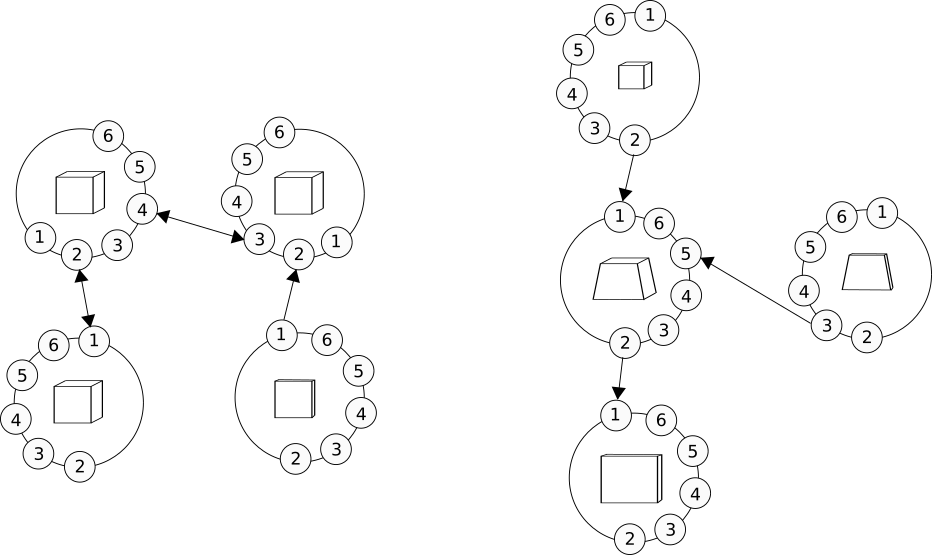


Figure . Graph representations of the buildings from Figure 79

Figure 79 presents examples of simple reference buildings, while Figure 80 shows their graph representations. The end-to-side and end-to-end relations between components are represented by the directed and undirected graph edges, respectively.

The components of the reference buildings may be grouped by the designer into patterns, as shown in Figure 81, where a pattern is marked with a green color and described as “WINDOW”. In this case, hierarchical nodes representing patterns are created in corresponding CP-graphs. Figure 82 contains a graph representation of the buildings from Figure 81.

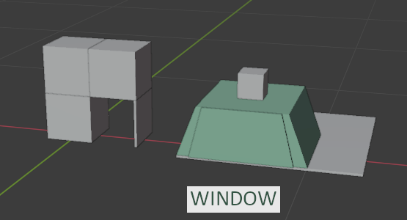


Figure . Components grouped into a pattern

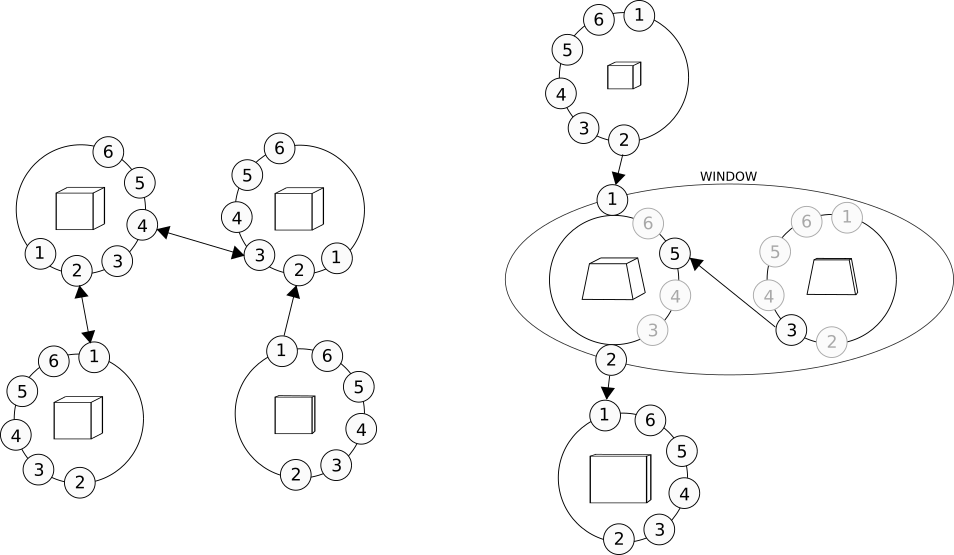


Figure . Graph with a hierarchical node representing a pattern

The patterns cannot nest and overlap, i.e., a component cannot belong to more than one pattern.

### Constructing the graph grammar

After CP-graphs of the reference buildings are created, the construction of the graph grammar takes place. Shortly speaking, for each node of the graph instance representing a reference building a graph grammar rule of adding this node is constructed. This requires the following steps:

1. For each graph instance , its nodes are traversed to find the minimal value of the z-coordinate assigned to a metric attribute “origin location”. This way the bottom components of a building are determined, as shown in Figure 83.

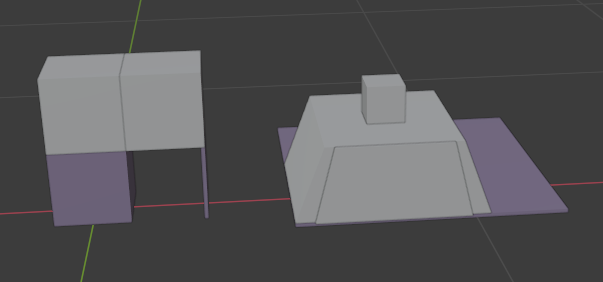


Figure . Bottom components of the reference buildings

1. For every found node :
   1. if belongs to any hierarchical node , then ,
   2. otherwise .
2. is then used to construct a graph grammar axiom. Let be a graph instance composed of a node isomorphic to in which the values of all the metric attributes have been reset. If does not contain an axiom isomorphic to , is added to the list of axioms of .

Figure 84 contains three axioms obtained from the graphs representing the reference buildings in Figure 83. In this case, each axiom is composed of a single node.

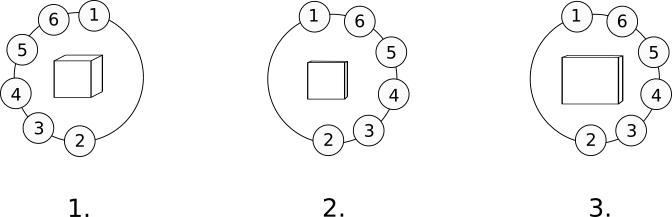


Figure . The axioms of the graph grammar based on the reference buildings from Figure 83

1. is marked as visited.
2. Now, for each node the incoming and outgoing edges of are considered. For each edge such that and is a node that has not yet been visited, a subgraph of is determined that contains , , and . Then, a rule is constructed, where:
   1. is a CP-graph instance composed of one node isomorphic to,
   2. is a CP-graph instance isomorphic to ,
   3. the values of all the metric attributes of and are reset.
3. is marked as visited and, if does not exist in , is added to .
4. If the node in corresponding to the single node of contains bonds with incoming or outgoing edges and are indexed with numbers greater than (i.e., represent the side surfaces of the component), for each such edge the analogical rules are created as described in 1.17.3.
5. and the procedure described in 5-8 is repeated recursively until all the nodes are visited.
6. If the parameter for defining reverted rules is set to true, for each rule in the reverted rules are created as described in 1.17.4.
7. If the parameter for defining modified rules is set to true, for some random rules of in which a single node is added the modified rules are created. This means that the grade of difference between the added component of the original rule and the one of the modified rule is in the range of (see section 1.17.1.2).

Figure 85 presents a graph grammar created on the basis of the CP-graphs from Figure 80. The added nodes are marked in bold. The analogical rules of the grammar are shown separately in Figure 86, while the reverted rules are in Figure 87. For clarity, the repeated rules as 3 and 4’ have not been removed from the picture, but it should be noticed that only one of them would be finally added to the grammar.

Figure 88 contains basic rules of graph grammar based on the graphs in Figure 82. Rule 5 allows attaching a hierarchical node representing a pattern, while rule 6 enables attaching another node to the hierarchical node. The list of axioms of the presented grammar is the same as in the one presented in Figure 84.

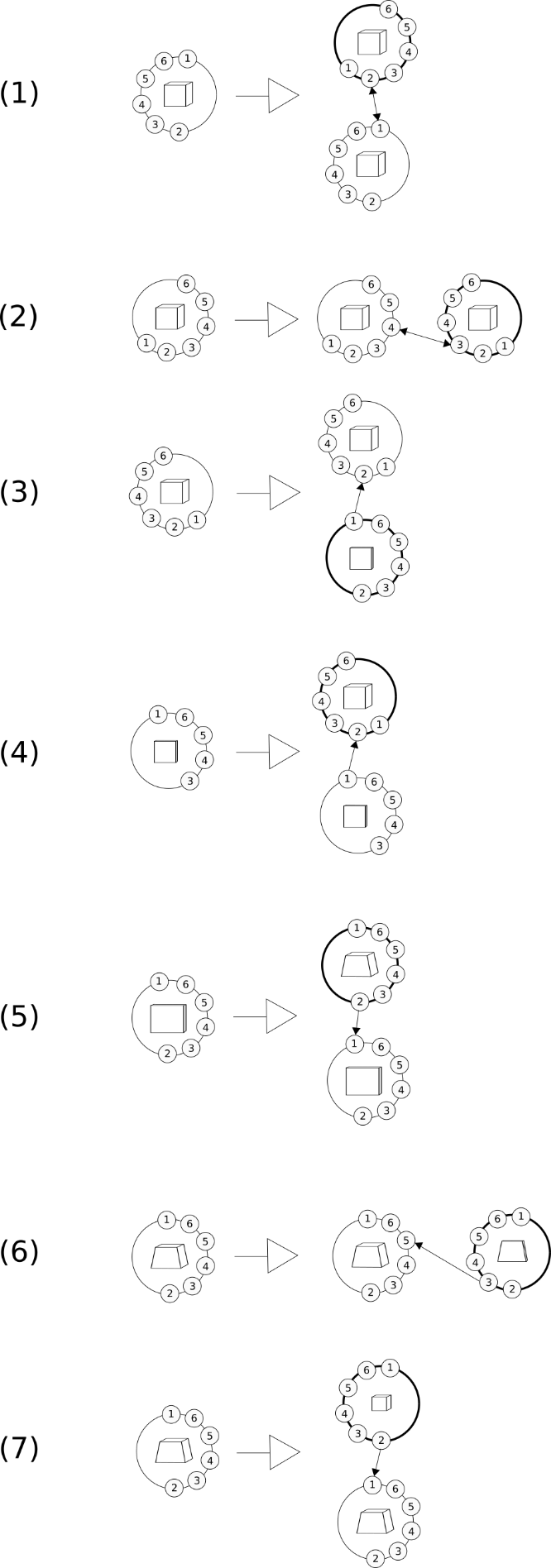


Figure . Graph grammar created on the basis of the CP-graphs from Figure 80

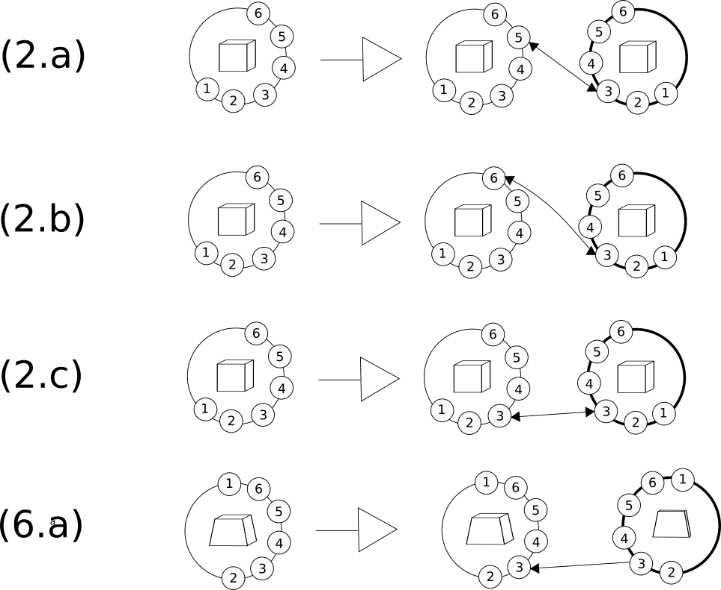


Figure . Analogical rules for the grammar in Figure 85

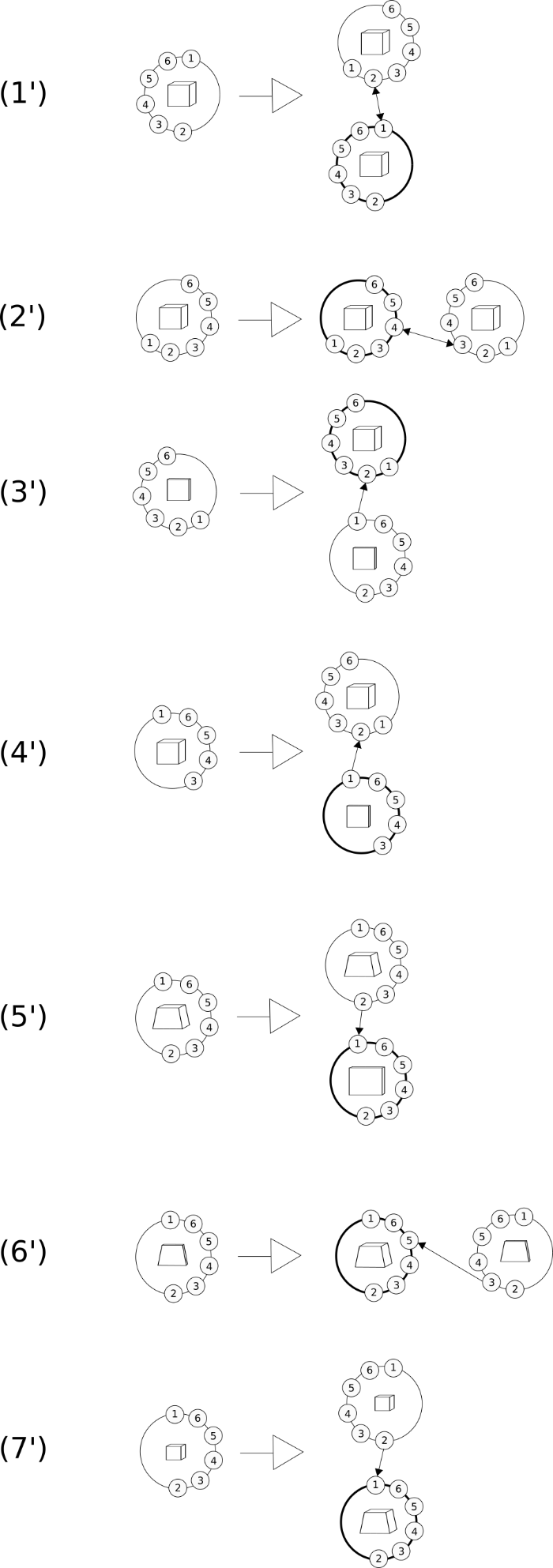


Figure . Reverted rules for the grammar in Figure 85

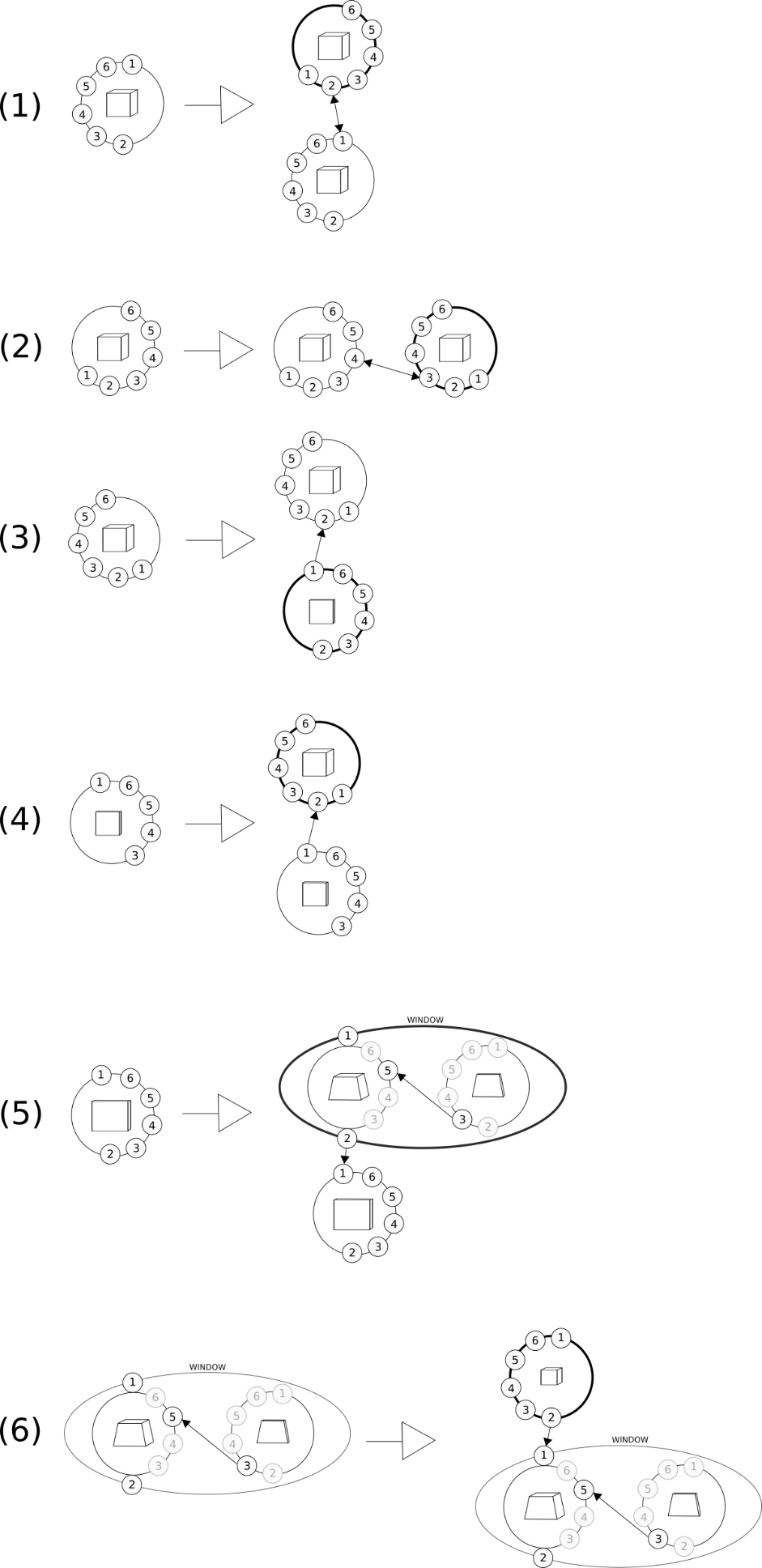


Figure . Graph grammar created on the basis of the CP-graphs from Figure 82

### Generating the initial population

Once the graph grammar is created, it can be used to generate a CP-graph instance representing a building. The generation is performed as follows:

1. An axiom of is randomly chosen and added to .
2. The values of the metric attributes of each node and bond in are randomly chosen within the range defined by the non-accidental attributes. In case of two components in the end-to-end relation it must be ensured that the adjacent surfaces are identical, i.e., the metric attributes of corresponding bonds have the same values.
3. A node is chosen randomly from until there exists a rule such that is composed of a node isomorphic to or until all the nodes from have been checked. If exists, is applied in on node , as described in section 1.3.
4. If exists, the values of the metric attributes of the added node and its contents are chosen analogically as in step 2.
5. Steps 1-4 are repeated until the required number of nodes is reached or there are no rules possible to apply.

The initial population for the genetic algorithm consists of individuals generated in the presented way. The number of individuals depends on the software parameter. Figure 89 contains some phenotypes of the initial population generated with the use of the basic rules of the grammar in Figure 85. For comparison, the buildings in Figure 90 have been generated with the grammar from Figure 88. It can be seen that in this case, it has not been possible to generate a trapezoidal solid without the window, as they form a pattern together.

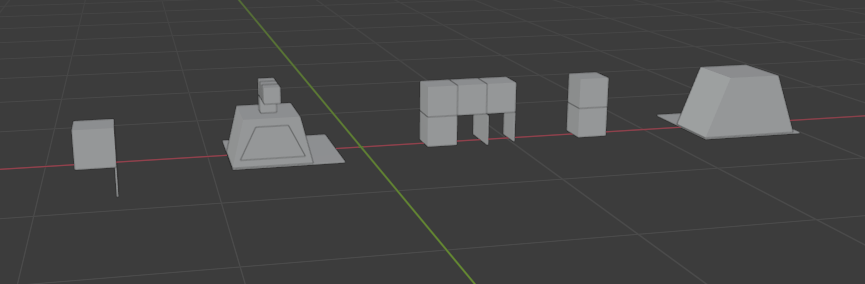


Figure . Example of an initial population based on the grammar in Figure 85

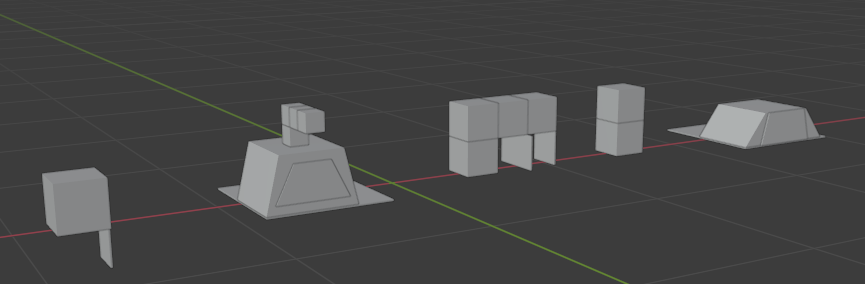


Figure . Example of an initial population based on the grammar in Figure 88

Figure 91 and Figure 93 contain more complex examples of buildings with defined patterns. Some of the buildings generated with the use of the obtained graph grammar are shown in Figure 92 and Figure 94, respectively.

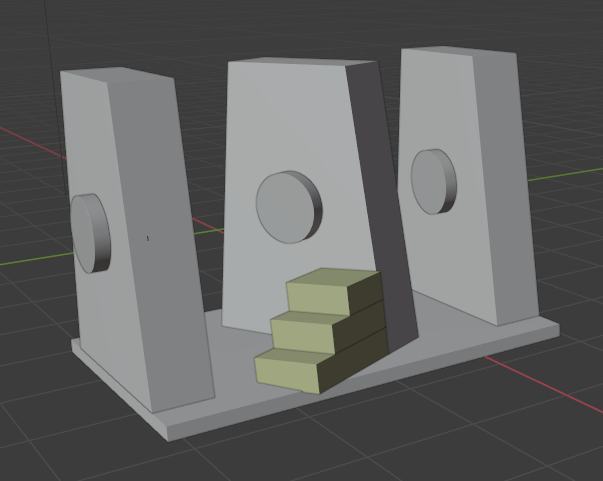


Figure . Reference building with a pattern of stairs

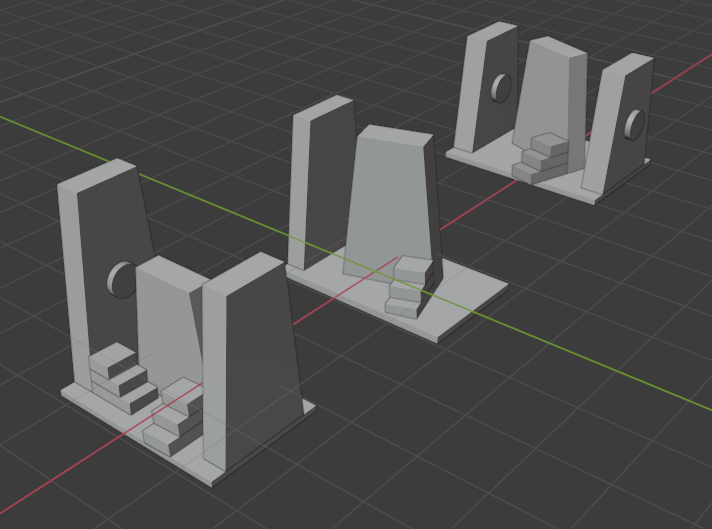


Figure . Initial population based on the building from Figure 91

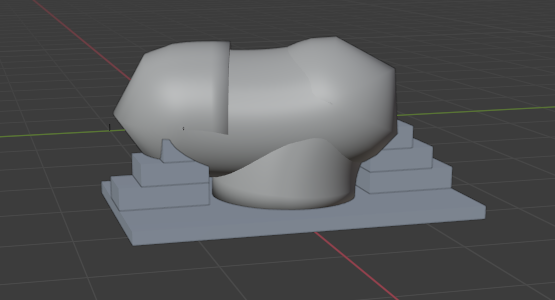


Figure . Reference building with a pattern of double stairs and a plinth

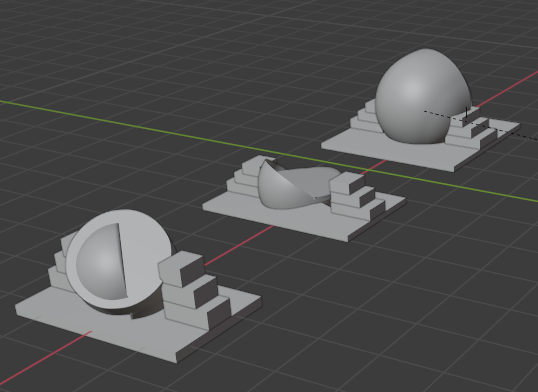


Figure . Initial population based on the building from Figure 93

## The genotype

The genotype of an individual is a sequence of the graph grammar rules applied to obtain a CP-graph representing the given building.

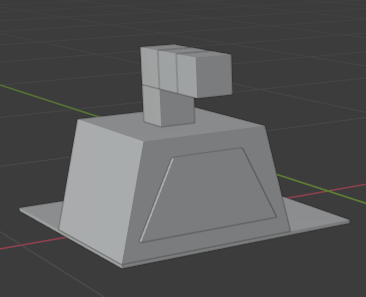


Figure . Example of a phenotype (1)

Let us consider a phenotype in Figure 95, which is a member of the initial population shown in Figure 90. Its graph representation has been generated by applying the following sequence of rules of the grammar presented in Figure 88: . Figure 96 shows the corresponding phenotype developing while applying the grammar rules to its CP-graph. The first version of the building is the visualization of the third axiom of the graph grammar (Figure 84), the further versions are obtained by applying the next rules from the rule sequence.

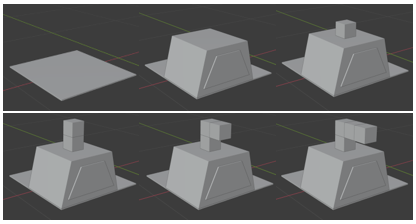


Figure . Phenotype obtained with a sequence of graph grammar rules

It can be noticed that a sequence of rules is not sufficient to fully represent an individual as it lacks information about subgraphs to which each rule should be applied. Therefore, each node, after being added to the constructed CP-graph, is indexed with the number of nodes representing a component of a given subtype that already exist in the graph. In case of adding a hierarchical node, it is indexed with the number of nodes with a given label. After that, the values of the metric attributes assigned to the added nodes and bonds are randomly chosen within the ranges defined by the non-accidental attributes. The genotype is then constructed as a tuple , where is a list of values of the metric attributes assigned to the grammar axiom used a starting graph In the genotype and is a sequence of triples , where:

1. is an index of a node in the graph corresponding to the left side of the applied rule, or, in the case of the first rule in the genotype, the index of the grammar axiom,
2. is the applied rule,
3. is a list of values of the metric attributes assigned to the new nodes and bonds added as result of applying on the node indexed with .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule number | (5) | (6) | (1) | (2) | (2) |
| Apply to | Axiom 3. | 1 | 1 | 2 | 3 |
| Result | 1 | 1 | 2 | 3 | 4 |

Table . The genotype of the building in Figure 95

Table 1 contains the genotype of the building in Figure 95. For clarity, the lists of metric attributes have been omitted and instead, the indexes of nodes added in each rule application are specified. At first, rule is applied to axiom of the graph grammar, which results in adding a hierarchical node “WINDOW” representing a pattern. This node is indexed with , because it is the first occurrence of “WINDOW” in the generated graph. The values of the metric attributes of the bonds and both nodes assigned to the hierarchical node, as well as of the bonds assigned to them are chosen. Then, rule is applied to the “WINDOW” node indexed with . This results in creating a node representing a cube and indexed with , as this is the first occurrence of this component subtype in the graph. Again, the values for the metric attributes of the added node and its bonds are assigned. Rule is then applied to the cube and cube is obtained. Further, rule is applied to the cube . Cube is added to the graph and rule is applied to it, which gives cube .

The presented method of encoding individuals as a sequence of grammar rules resembles binary encoding and has its advantage in the simplicity of performing the genetic operations.

Figure 97 presents another phenotype represented by a graph generated by the same graph grammar as in the previous example. Table 2 contains the genotype of the discussed individual.

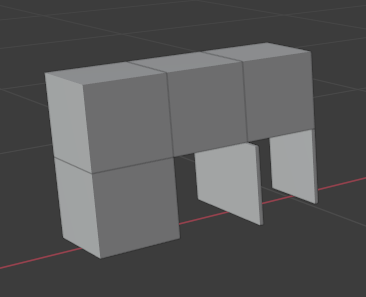


Figure . Example of a phenotype (2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule number | (1) | (2) | (4) | (2) | (4) |
| Apply to | Axiom 1. | 1 | 2 | 2 | 3 |
| Result | 1 | 2 | 1 | 3 | 2 |

Table . . The genotype of the building in Figure 97

## The fitness function

The fitness function aims to assess how much the aesthetics of the derivative building resembles the aesthetics of the reference buildings. Therefore, the applied fitness function uses the formula described in section 1.18.4 “Aesthetic measure for comparison of architectural forms”. Additionally, collisions of components are detected and deprecated since the application of grammar rules in the genotype sequence may result in overlapping components. Although some interesting forms may emerge this way, full overlapping brings no benefits to the final result and increases the cost of the evaluation. Therefore, the fitness function uses the following formula for evaluation: , where is calculated as presented in section 1.18.4, and is equal to in case of detecting fully overlapping components in the building, and to in the opposite case.

## The selection

In the presented system each step of the evolution is presented to the designer, who can preview the individuals along with the values of their evaluation. For this reason, the author has resigned from any advanced form of selection, as it might confuse the user by allowing reproduction of poorly adjusted individuals. Instead, the truncation selection has been chosen.

## The crossover

The crossover operator uses two parent genotypes in the form of rules sequences to generate two descendant genotypes. Tor each parent, the center of the sequence is determined and the genotype is cut into two halves in this place (in case of an odd number of rules in a sequence, the first cut part contains one rule more than the second part). This process is presented in Table 3 and Table 4, where the individuals from Figure 95 and Figure 97 are taken as parents.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rule number | (5) | (6) | (1) |  | (2) | (2) |
| Apply to | Axiom 3. | 1 | 1 | 2 | 3 |
| Result | 1 | 1 | 2 | 3 | 4 |

Table . Division of the first parent genotype

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rule number | (1) | (2) | (4) |  | (2) | (4) |
| Apply to | Axiom 1. | 1 | 2 | 2 | 3 |
| Result | 1 | 2 | 1 | 3 | 2 |

Table . Division of the second parent genotype

After that, the first part of the first parent is concatenated with the second part of the second parent, as in Table 5. This way the first child genotype is constructed. To obtain the genotype of another child, the first part of the second parent is concatenated with the second part of the first parent, as presented in Table 6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule number | (5) | (6) | (1) | (2) | (4) |
| Apply to | Axiom 3. | 1 | 1 | 2 | 3 |
| Result | 1 | 1 | 2 | 3 | 2 |

Table . Genotype of the first child

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule number | (1) | (2) | (4) | (2) | (2) |
| Apply to | Axiom 1. | 1 | 2 | 2 | 3 |
| Result | 1 | 2 | 1 | 3 | 4 |

Table . Genotype of the second child

The advantages of this approach to the crossover are its simplicity and the fact that the obtained graphs belong to the language generated by the same graph grammar as the graphs from the initial population, which helps to preserve the aesthetic character of the evolved individuals.

The main difficulty in the presented method of the crossover is a situation when the descendant sequence contains rules that ought to be applied to the nodes which are not present in the graph. In this case, the application of such a rule is omitted. It can also happen that nodes representing a given component subtype or a pattern are present in the graph, but there is no node indexed with the required number. In this case, the node with the closest index is chosen.

Figure 98 and Figure 99 present phenotypes of individuals encoded by the genotypes in Table 5 and Table 6, respectively.

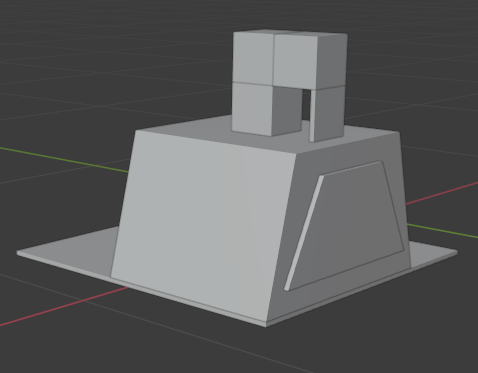


Figure . Phenotype of the first child

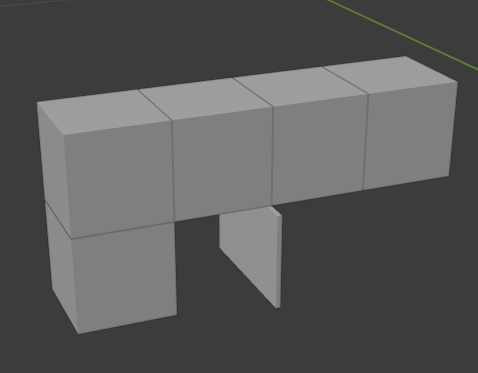


Figure . Phenotype of the second child

## The mutation

There are the following variants of the mutation operator:

1. The modification of a metric attribute value.

For randomly chosen metric attributes, new values are assigned. This modification may regard any node added to the graph during the application of the rule sequence or the grammar axiom used in the genotype.

Figure 100 presents a result of the axiom modification – the width and height assigned to the axiom node of the genotype shown in Table 5 have been altered.

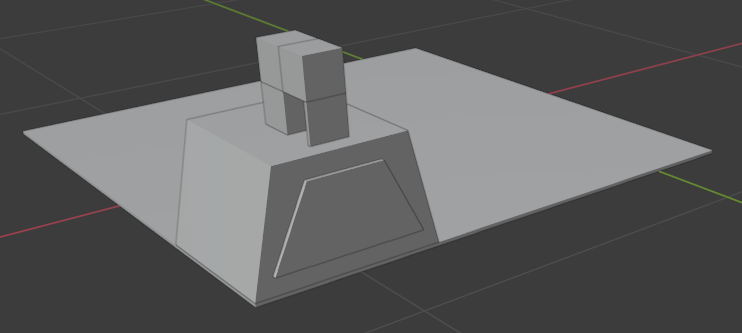


Figure . Mutation by altering metric attributes’ values – the resultant phenotype

1. The removal of a rule in the rule sequence.

A randomly chosen rule in the genotype is removed. This may result in the problems described in “The crossover” section – the following rules in the sequence may not be possible to apply. Again, such rules are omitted in the process of graph construction. In case of the lack of the appropriate index of the graph node, the closest one is chosen to apply the rule.

Table 7 presents a genotype from Table 6 in which the second rule in the sequence has been deleted. Figure 101 contains the resultant phenotype. It can be noticed that in this case, the mutation is definitely harmful, as the obtained building lacks stability.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rule number | (1) | (4) | (2) | (2) |
| Apply to | Axiom 1. | 2 | 2 | 3 |
| Result | 1 | 1 | 3 | 4 |

Table . Mutation by removing a rule – the resultant genotype

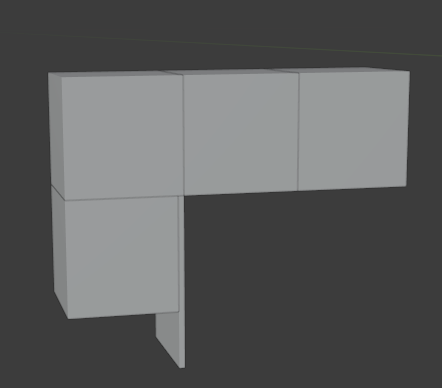


Figure . Mutation by removing a rule – the resultant phenotype

1. The addition of a rule into the rule sequence.

A random grammar rule is inserted in a random place in the rule sequence. The index of the node to which the rule should be applied is chosen randomly from the corresponding nodes added earlier in the sequence. If no appropriate node exists, the index is set to and the rule cannot be applied.

Table 8 presents such a modification of the genotype from Table 6. Rule has been inserted after the fourth rule of the sequence and applied to the cube indexed with .

The resultant phenotype is shown in Figure 102.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rule number | (1) | (2) | (4) | (2) | (1) | (2) |
| Apply to | Axiom 1. | 1 | 2 | 2 | 2 | 3 |
| Result | 1 | 2 | 1 | 3 | 4 | 5 |

Table . Mutation by adding a rule – the resultant genotype

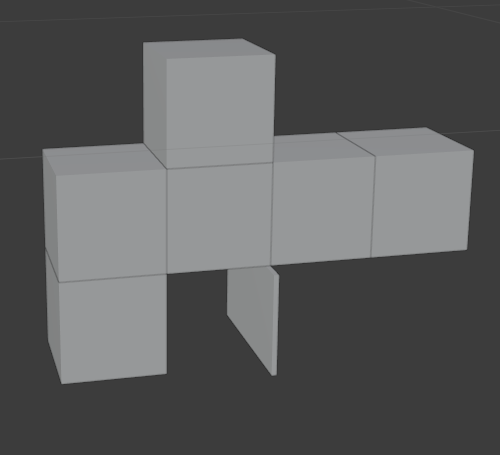


Figure . Mutation by adding a rule – the resultant phenotype

## The termination condition

In the presented approach, the evolution is led by the designer who chooses to create the next generation of designs. When the designer decides that the results are satisfactory enough they can terminate the program and save the selected designs.

# Chapter 8. The Results

## Examples of program results

Some results of the presented genetic algorithm are shown in the following figures. Figure 103 contains a reference building whose aesthetic components are evaluated as shown in Figure 104. In Figure 105, there is an example of the initial population derived from this reference building. The system parameter “Reverted rules” has been set to “False”. Three buildings are presented, with their aesthetic evaluation shown in Figure 106. The field “Final value” in Figure 106 refers to the fitness function of the algorithm, which calculates to what degree the aesthetic character of the evaluated building differs from the aesthetic character of the reference building. Figure 107 shows buildings generated after fifty iterations of the genetic algorithm, with their evaluation in Figure 108.

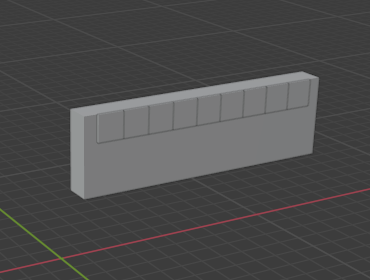


Figure . Reference building with high values of alignment and mirror symmetry

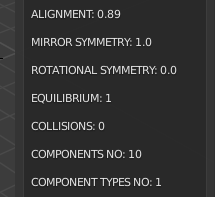


Figure . Evaluation of the building from Figure 103

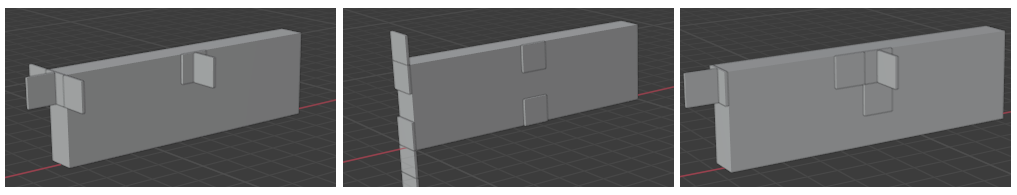


Figure . Buildings from the initial population generated on the basis of the building in Figure 103

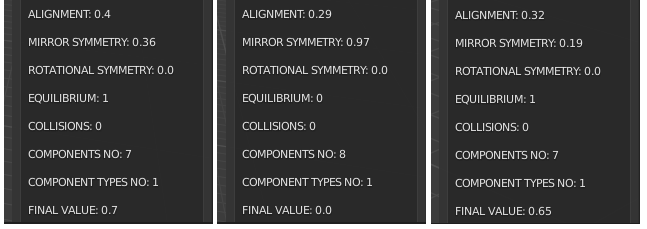


Figure . Evaluation of the buildings from Figure 106

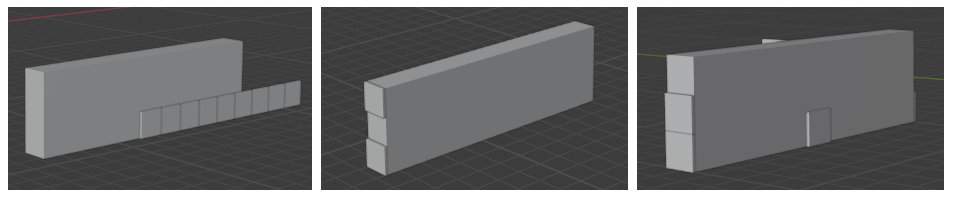


Figure . Evolution results for the reference building showed in Figure 103

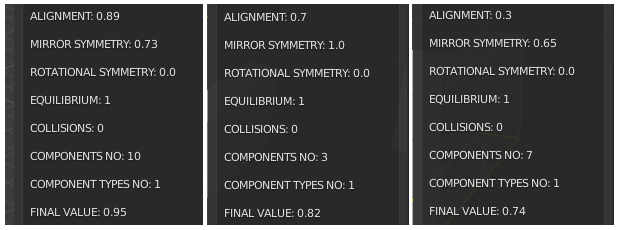


Figure . Evaluation of the buildings from Figure 107

Another example of the initial population based on the reference buildings from Figure 103 is shown in Figure 109, while Figure 110 contains the evaluation of aesthetic components of the generated buildings. This time the parameter “Reverted rules” has been set to “True”. The results after fifty iterations are presented in Figure 111, and their evaluation is shown in Figure 112.

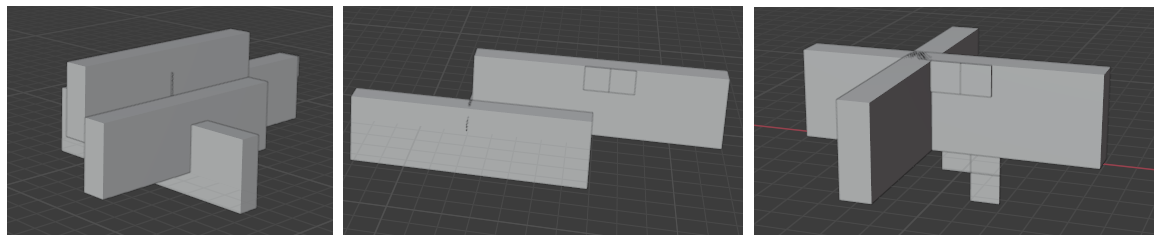


Figure . Another initial population generated on the basis of the building in Figure 103 (with the use of graph grammar reverted rules)

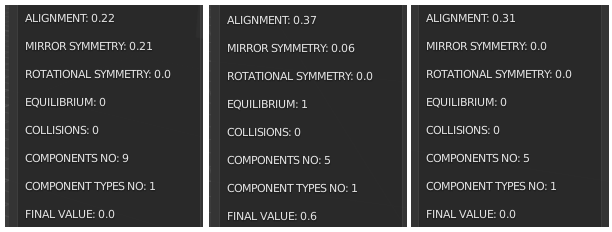


Figure . Evaluation of the buildings in Figure 109

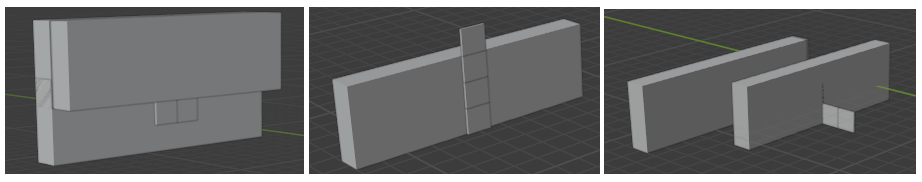


Figure . Evolution results for the reference building shown in Figure 103 (with the use of graph grammar reverted rules)

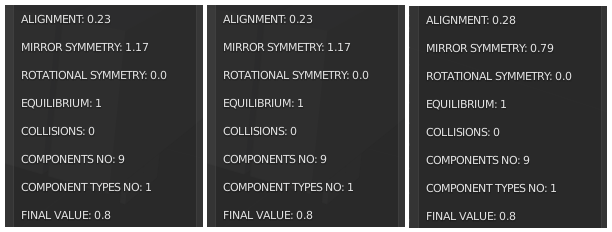


Figure . Evaluation of the buildings in Figure 111

Figure 113, Figure 116 and Figure 119 contain other examples of reference buildings. Figure 114, Figure 117 and Figure 120 present generated initial populations, while Figure 115, Figure 118 and Figure 121 present the results evolved in response to each of these examples.

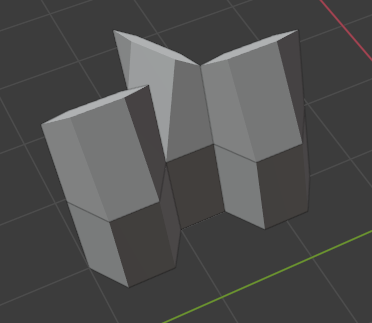


Figure . Reference building with high value of mirror symmetry

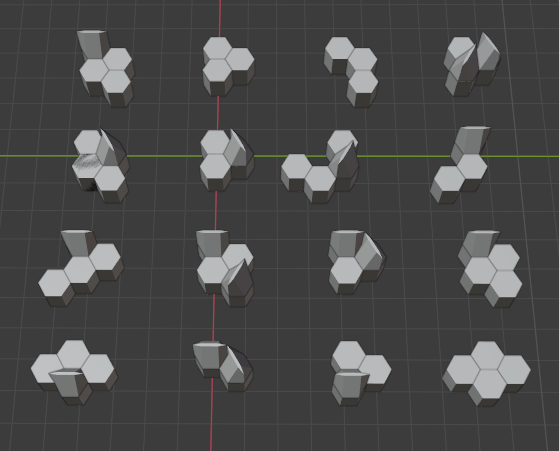


Figure . Initial population based on the building from Figure 113

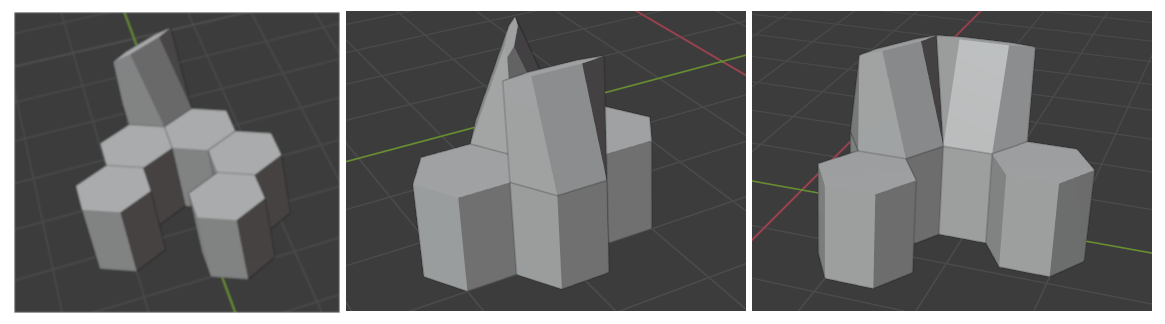


Figure . Evolution results for the reference building from Figure 113

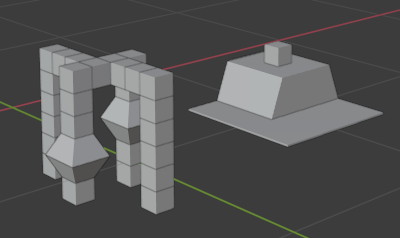


Figure . Two reference buildings

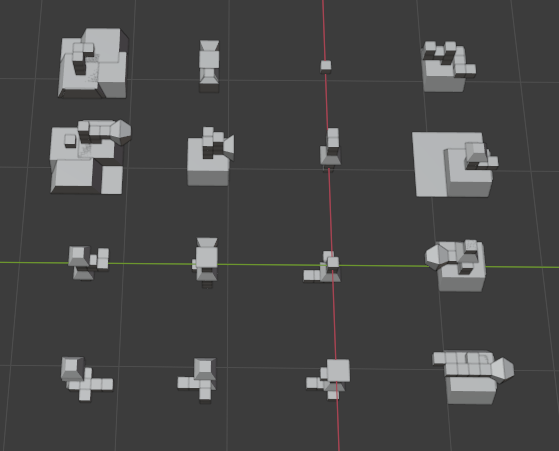


Figure . Initial population based on the buildings from Figure 116

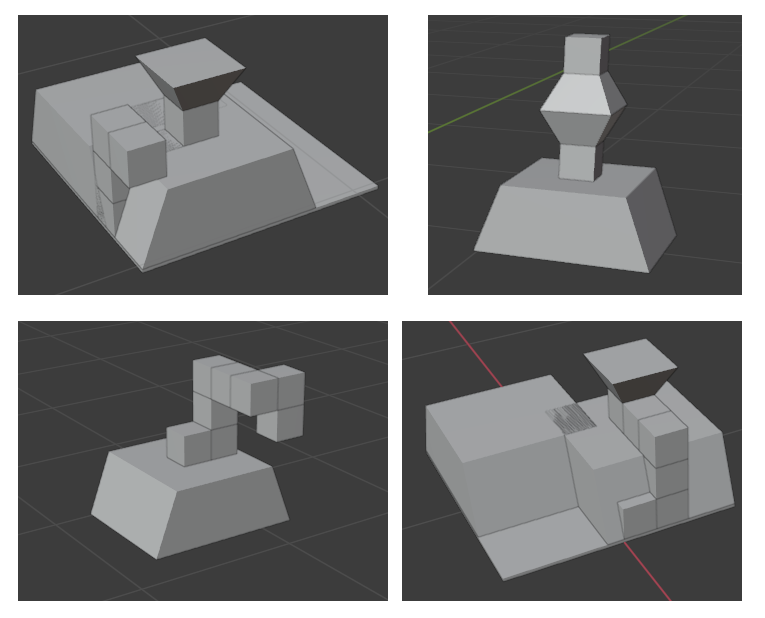


Figure . Evolution results for the reference buildings from Figure 116

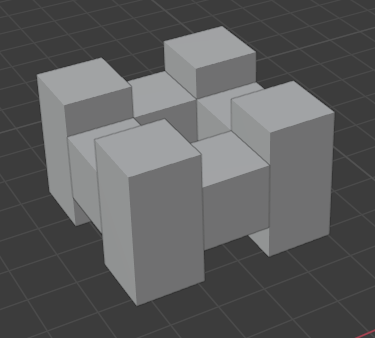


Figure . Reference building with high values of both rotational and mirror symmetry

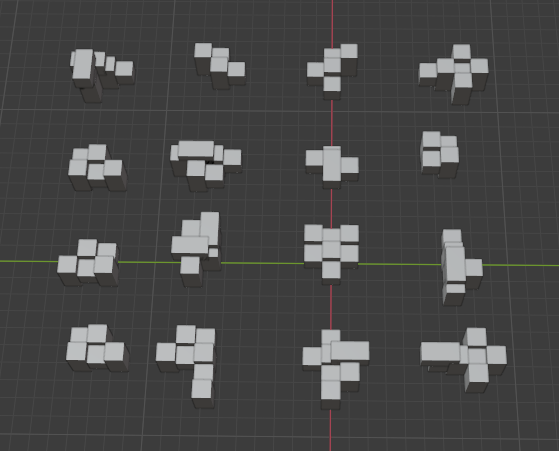


Figure . Initial population based on the building from Figure 119

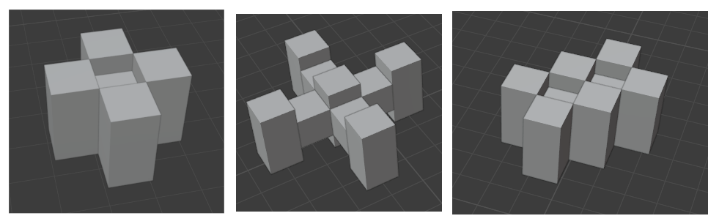


Figure . Evolution results for the reference building from Figure 119

## Summary

In this dissertation, the author has proposed a design method that implements the Recognition-By-Components model of Biederman. Architectural forms are created by the designer with the use of basic components corresponding to Biederman’s geons. In the presented solution composition graphs have been applied and adapted for internal representation of designs, and graph grammars have been employed to generate buildings of similar aesthetic character to the reference building provided by the designer. Further, the genetic algorithm has been proposed and implemented for the optimisation of aesthetic features corresponding to the reference building.

The presented approach enables the generation of simple buildings in a given aesthetic character. The design process is to a large extent driven by the designer, who, except from providing the reference building, can set the values of parameters of the constructed graph grammar and of the evolution process. The designer may also modify individuals in the given generation and decide when the evolution stops.

The genotype in a form of a sequence of graph grammar rules resembles, by its simplicity, the binary encoding while enabling fast translation into the graph representation. The crossover operator is very easy to implement on a sequence of rules, contrary to the crossover of graphs, which is a more complex problem. The descendant forms belong to the language generated by the graph grammar that is used to generate the initial population, which helps to preserve the aesthetic character of the designs.

The obtained results show the increase of the aesthetic similarity between the population individuals and the reference buildings, with aesthetic components like mirror and rotational symmetry, alignment, number of components, number of component types, equilibrium and number of solid collisions taken into account.

Of course, the presented approach has some limitations. The Biederman model assumes the perceived objects can be divided into parts possible to be mapped into simple abstract solids (geons), which is not always true, especially in the case of organic forms. What is more, this model distinguishes only two main relations between the components, “end-to-end” and “end-to-side”. The relations indicating some distance between the components, like „above”, “behind’, etc. are not regarded, which limits the possibilities of architectural design.

Another limitation is the usage of a simple context-free graph grammar to generate the initial population. On the one hand, this approach enables the effective generation of graphs. On the other, it prevents obtaining graphs in which the right side of the applied rule is connected to more than one node of the developed graph.

## Further work

Further work will include attempts to overcome the stated limitations. More relations can be considered to construct more complex forms. The concept of design patterns will be developed by enabling nested hierarchies and dynamic construction of patterns, where each pattern has assigned a separate graph grammar that generates similar forms. This can result in more advanced possibilities for graph embedding.

Another field of research is the aesthetic measure applied in the fitness function. At the moment, it regards only perfect relations of alignment and symmetry, which corresponds to Birkhoff’s approach, in which imperfect relations of order are perceived negatively. This conception does not seem to include organic forms, where perfect symmetry and alignment relations rarely occur, but it does not prevent the perceived object from being regarded as beautiful.

# Appendices

## Appendix A: The algorithms for determining the number of order relations

### The algorithm for determining the number of alignment relations in a group of components

Input: a list of components  
Output: a list of numbers

The number of alignment relations is computed in the following steps:

1. For each building component , for each face of , a plane such that is found.
2. The equation of is normalized, which enables the comparison of different planes.
3. A signed distance between and the center of mass of is calculated. If , a variable . Otherwise, . This step is necessary to distinguish between the components aligned to the same plane, as in Figure 122, and the ones attached to each other as in Figure 123. In the first case, the values of both components are the same, while in the second case, they are opposite.

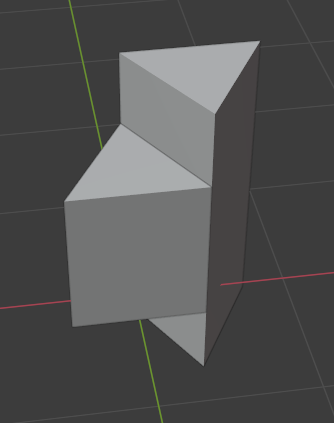


Figure . Components attached to each other on the different sides of the same plane

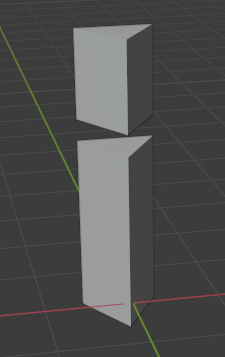


Figure . Components aligned to the same plane

1. A tuple is constructed as a key for a dictionary .
2. If does not exist, a new entry is added to . Otherwise, the value of is increased by .

As result, contains a set of different positive and negative sides of different planes, for each of whom a number of aligned faces is stored. The algorithm returns a list of values of .

### The algorithm for determining the number of partial mirror symmetries in a group of components

Input: a list of components  
Output: a list of lists of components

The number of partial mirror symmetries is determined as follows:

1. For each pair of components , , the corresponding non-accidental and metric properties of and are compared. If their values are identical, .
2. If :
   1. A plane equidistant to the mass centers of and is found and normalized.
   2. For every vertex in , a distance between and is calculated. After that, the vertices of are traversed in order to find a vertex distant from by .
   3. If for every a corresponding vertex exists, and are appended to the component list of the entry of a dictionary whose key is .

The dictionary contains now the planes of the partial symmetries associated with a list of components arranged on both sides of each plane. One needs now to find components that are intersected by the given planes and symmetrical about them.

1. For each entry in , traverse the building components. For each component verify its symmetry about :
   1. For each vertex in find its distance from and search for another vertex whose distance from equals and a line determined by and is perpendicular to . If is found, mark and as symmetrical.
   2. If all the vertices of are marked as symmetrical, is regarded as symmetrical about and appended to .

The algorithm returns a list of numbers of components in the found relations obtained from the list of values of .

### The algorithm for determining the number of partial rotational symmetries in a group of components

Input: a list of components  
Output: a list of lists of components

The number of partial rotational symmetries is found in the following steps:

1. For each pair of components , with the centers of mass and , respectively, the distance between and is found.
2. The pairs of centers of mass are grouped by their value into distance groups. Each distance group contains then a set of unordered point pairs .
3. For each distance group , a set of triangles is searched for such that for each defined by a triple of vertices , and .
4. For each triangle , a plane containing the three points of is found and normalized analogically as in the alignment value computation (1.18.1). The set is then divided into subsets of triangles belonging to the same plane.
5. Each triangle subset is processed as follows until there are any triangles remaining in :
   1. The first triangle is moved from into the potential symmetry list and copied as , and is calculated.
   2. The next triangle such that is searched for in .
   3. If does not exist, is cleared and the algorithm continues in 5a. Otherwise, the two cases are possible:
      1. . is then moved from into and its vertices replace the ones of , i.e., , and . The algorithm then continues in 5b.
      2. , which means that a regular polygon is found. Its center of mass is calculated and a symmetry axis such that and is determined. After that, a tuple , where denote the regular polygon vertices stored in is constructed. is then appended to the dictionary of symmetries at the end of tuple list . is cleared and the algorithm continues in 5a.

In result of applying 5 to all the triangles found for every plane and every distance, the dictionary contains all the regular polygons found among the centers of mass of the building components, assigned to their symmetry axes.

The steps presented in 1-5 contain an adaptation of the algorithm described in [125]. Further steps contain the extension of that algorithm.

At first, it should be noticed that a single entry of contains a set of polygons which, although symmetrical against a common axis, do not necessarily form a symmetric figure together. Figure 124 and Figure 125 present this problem reduced to the two-dimensional space. Figure 124 contains two pairs of regular polygons. Despite common centers of rotational symmetry, the figures formed by these pairs are not symmetrical. Another situation is shown in Figure 125, where the formed figures have rotational symmetry. One can observe that rotational symmetry in a figure composed of a set of regular polygons occurs only if for each pair of -gon and -gon belonging to , or , where . This conclusion can be transferred into a three-dimensional space, where the considered polygons are symmetrical against a common axis.

For further discussion, let us a call composition of regular polygons that has rotational symmetry a *multipolygon*.

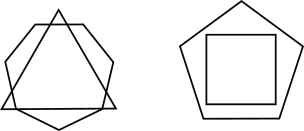


Figure . Lack of rotational symmetry in polygon groups

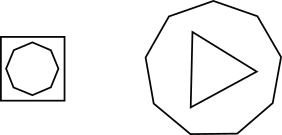


Figure . Rotational symmetry in polygon groups

1. For each entry in , *multipolygons* of are determined. After that is divided into sublists, each containing *multipolygon* vertices.

Now, each polygon needs to be examined for rotational symmetry in the components it gathers.

1. For each entry in , for each *multipolygon* and for each , each pair of neighbouring vertices is analyzed:
   1. Components , whose centers of mass are the points of the analyzed pair are compared. If their non-accidental and metric properties are identical, . Otherwise, is removed from .
   2. If , for each vertex , a corresponding symmetrical vertex in is searched for:
      1. For every , is calculated, where and are projections of and onto . If ,
      2. If is not found, is removed from .
   3. If every vertex in has a corresponding vertex in , and are regarded as parts of the symmetry.
2. If every component corresponding to is a part of symmetry, the component group is symmetrical with the symmetry axis . In this case, is searched for elements in which a list of corresponding components is included in . Such elements are removed from . This step eliminates additional rotational symmetries found in a group of regular polygon vertices – e.g., the vertices of a regular hexagon contain vertices of two equilateral triangles, which should not be considered in the overall result.
3. As stated in [125], each symmetry stored in may contain additional central component. In order to find it, for each axis in the building components are traversed. If a center of mass of component belongs to , is considered as a potential part of symmetry and analyzed as follows:
   1. For each triple of vertices a plane is found and its equation is normalized. If , the vertices are appended to the dictionary of planes at the end of vertices list .
   2. For every vertex of that is unassigned to , it is checked if . If the condition is false, is considered as not symmetrical.
   3. Otherwise, for each plane in
      1. The assigned vertices are analyzed analogically as in 1-3, 5 to find regular polygons.
      2. If there are vertices of in not belonging to regular polygons, is considered as not symmetrical.
   4. If all the vertices of all the planes from form regular polygons, they are checked for being a *multipolygon*, analogically as in 6. If all the verified vertices form a *multipolygon* , is considered as symmetrical. Otherwise, it is not symmetrical.
   5. If is symmetrical, for each *multipolygon* , if forms a *multipolygon*, a center of mass of i s appended to .

The algorithm returns a list of numbers of components in the found relations obtained from the list of values of .

## Appendix B: The source code of the genetic algorithm

class WM\_EVOLUTION:

wm\_evolution\_call\_counter = 0

mutation\_probability = 0.2

def AddGraphToPopulation(self, \_graph\_grammar, rules):

graph\_id = -1

if rules is not None and len(rules) > 0:

graph\_id = \_graph\_grammar.append\_empty\_graph\_to\_population()

erl\_object = \_graph\_grammar.get\_erl\_object\_by\_id(graph\_id)

erl\_object.append\_rules(rules)

\_graph\_grammar.recover\_graph\_from\_erl\_list(graph\_id, erl\_object, '.')

graph = \_graph\_grammar.get\_graph\_by\_id(graph\_id)

else:

utils.log("Empty rule list.")

return graph\_id

def Crossover(self, \_graph\_grammar, \_parent1GraphID, \_parent2GraphID):

parent1\_rules = \_graph\_grammar.get\_erl\_object\_by\_id(\_parent1GraphID).get()[1]

parent2\_rules = \_graph\_grammar.get\_erl\_object\_by\_id(\_parent2GraphID).get()[1]

div1 = int(len(parent1\_rules) / 2)

div2 = int(len(parent2\_rules) / 2)

child1\_rules = parent1\_rules[:div1]

child1\_rules.extend(parent2\_rules[div2:])

child2\_rules = parent2\_rules[:div2]

child2\_rules.extend(parent1\_rules[div1:])

child1\_id = self.AddGraphToPopulation(\_graph\_grammar, child1\_rules)

child2\_id = self.AddGraphToPopulation(\_graph\_grammar, child2\_rules)

def Recover(self, \_graph, \_graph\_grammar):

erl\_object = \_graph\_grammar.get\_erl\_object\_by\_id(\_graph.GetID())

\_graph\_grammar.recover\_graph\_from\_erl\_list(\_graph.GetID(), erl\_object, '.')

def ChangeMetricProperties(self, \_graph, \_graph\_grammar):

# Get ERL object reference

erl = \_graph\_grammar.get\_erl\_object\_reference\_by\_id(\_graph.GetID())

# Get number of items

erl\_num\_items = erl.get\_num\_items()

# Choose an item to modify size

erl\_i = random.randint(0, erl\_num\_items)

# Get metric properties (MP stored in array, for non group item it is one elements array)

mp\_object\_list = erl.get\_metric\_properties(erl\_i)

# Existance test

if mp\_object\_list is None:

utils.log("No mp\_object\_list for erl item " + str(erl\_i))

return

else:

utils.log("Found mp\_object\_list for erl item " + str(erl\_i))

mp\_item = mp\_object\_list[0]

# Choose one of WM\_Objects.modify\_size() arguments

param\_no = random.randint(1, 3)

if param\_no == 1: # Modify width

size\_modification = random.uniform(-1 \* mp\_item.bottom\_width + 1, mp\_item.bottom\_width)

mp\_item.modify\_size(\_bottom\_width = mp\_item.bottom\_width + size\_modification, \_top\_width = mp\_item.top\_width + size\_modification, \_regenerate\_hash=True)

elif param\_no == 2: # Modify length

size\_modification = random.uniform(-1 \* mp\_item.bottom\_height + 1, mp\_item.bottom\_height)

mp\_item.modify\_size(\_bottom\_height = mp\_item.bottom\_height + size\_modification, \_top\_height = mp\_item.top\_height + size\_modification, \_regenerate\_hash=True)

else: # Modify height

size\_modification = random.uniform(-1 \* mp\_item.height + 1, mp\_item.height)

mp\_item.modify\_size(\_height = mp\_item.height + size\_modification, \_regenerate\_hash=True)

# Store item in ERL

erl.set\_metric\_properties(erl\_i, mp\_item, False, 0)

erl\_pn\_hash, erl\_wm\_mp = erl.get\_pn\_hash\_and\_mp\_object(erl\_i)

mp\_update\_status = \_graph\_grammar.append\_metric\_properties\_to\_pure\_node(erl\_pn\_hash, erl\_wm\_mp)

if mp\_update\_status is False:

utils.log("[WM\_EVOLUTION: ChangeMetricProperties]: error in metric properties change")

def ChangeChromosomeStructure(self, \_graph, \_graph\_grammar):

rules = \_graph\_grammar.get\_erl\_object\_reference\_by\_id(\_graph.GetID()).get()[1]

last\_rule\_index = len(rules) - 1

if (last\_rule\_index < 1):

return

index = random.randint(1, last\_rule\_index)

type = random.randint(1, 3)

if (type == 1):

del rules[index]

elif(type == 2):

rule = rules[random.randint(1, last\_rule\_index)]

rules.insert(index, rule)

else:

rule = rules[random.randint(1, last\_rule\_index)]

del rules[index]

rules.insert(index, rule)

def Mutate(self, \_graph, \_graph\_grammar):

num = random.random()

if (num < self.mutation\_probability / 2):

self.ChangeMetricProperties(\_graph, \_graph\_grammar)

elif (num >= self.mutation\_probability / 2 and num <= self.mutation\_probability):

self.ChangeChromosomeStructure(\_graph, \_graph\_grammar)

self.Recover(\_graph, \_graph\_grammar)

# List of Figures

[Figure 1. Schema of the generative system xii](#_Toc137205195)

[Figure 2. Alternative representations of an apartment 2](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205196)

[Figure 3. Different views of an object 3](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205197)

[Figure 4. Objects composed from geons 4](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205198)

[Figure 5. Thirty-six geons as combinations of four non-accidental properties 5](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205199)

[Figure 6. Different types of geon termination 6](#_Toc137205200)

[Figure 7. Spatial relations between geons 7](#_Toc137205201)

[Figure 8. Size relations between geons 7](#_Toc137205202)

[Figure 9. Different values of the cross-section edge number property 8](#_Toc137205203)

[Figure 10. Detailed properties of each component type 10](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205204)

[Figure 11. Example of a NURBS curve 11](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205205)

[Figure 12. Control polylines of component curves 12](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205206)

[Figure 13. End-to-end relation between components 13](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205207)

[Figure 14. Center-to-center relation between components 13](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205208)

[Figure 15. Edge-to-edge relation between components 14](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205209)

[Figure 16. End-to-side relation between components 14](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205210)

[Figure 17. A design object 16](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205211)

[Figure 18. A labeled directed graph representing an object from Figure 17 16](#_Toc137205212)

[Figure 19. Different attachments between the same components 17](#_Toc137205213)

[Figure 20. A labeled directed graph representing the objects from Figure 19 18](#_Toc137205214)

[Figure 21. Two components with different cross section types – a curved one and a triangular one 19](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205215)

[Figure 22. CP-graph nodes representing components from Figure 21 19](file:///C:\Ag\inf\praca\rozdzialy\praca_doktorska_mars.docx#_Toc137205216)

[Figure 23. Labeled directed CP-graphs representing the buildings from Figure 19 20](#_Toc137205217)

[Figure 24. A CP-graph representing the object from Figure 17 21](#_Toc137205218)

[Figure 25. Example of a building with a user-defined pattern 22](#_Toc137205219)

[Figure 26. Graph representing the building from Figure 25 23](#_Toc137205220)

[Figure 27. Example of a simple CP-graph grammar 27](#_Toc137205221)

[Figure 28. Example of derivation by means of the grammar shown in Figure 27 28](#_Toc137205222)

[Figure 29. Buildings represented by the graph derived in Figure 28 29](#_Toc137205223)

[Figure 30. Urban scenery generated by CityEngine [52] 33](#_Toc137205224)

[b) Figure 31. Surprising results of applying a shape grammar rule 34](#_Toc137205225)

[Figure 32. Example of images generated by the Midjourney program 35](#_Toc137205226)

[Figure 33. Coffee pots generated by Celestino Soddu’s software Argenia [61] 36](#_Toc137205227)

[Figure 34. Genetic algorithm diagramm 44](#_Toc137205228)

[Figure 35. Examples of polygons evaluated by Birkhoff aesthetic measure (source: [9]) 47](#_Toc137205229)

[Figure 36. Villa Savoye designed by Le Corbusier [106] 50](#_Toc137205230)

[Figure 37. Hōryū-ji temple in Ikaruga [107] 51](#_Toc137205231)

[Figure 38. Mokoulek – a plane and a photograph of the village [111] 52](#_Toc137205232)

[Figure 39. *Szklany Dom* in Warsaw designed by Juliusz Żórawski [120] 54](#_Toc137205233)

[Figure 40. Architectural forms based on symmetry, generated by Gardner & Krishnamurti’s software (screenshots published with author’s permission) 56](#_Toc137205234)

[Figure 41. A building made from two types of components 58](#_Toc137205235)

[Figure 42. Buildings generated by the grammar obtained from the building in Figure 41 58](#_Toc137205236)

[Figure 43. Modification of metric properties of the buildings from Figure 42 59](#_Toc137205237)

[Figure 44. Buildings with different component types () 59](#_Toc137205238)

[Figure 45. Example of 60](#_Toc137205239)

[Figure 46. Example of 60](#_Toc137205240)

[Figure 47. Example of 61](#_Toc137205241)

[Figure 48. Buildings with different component types () 61](#_Toc137205242)

[Figure 49. Invalid production rule 62](#_Toc137205243)

[Figure 50. Building made from two components 63](#_Toc137205244)

[Figure 51. Grammar rule attaching components from Figure 50 63](#_Toc137205245)

[Figure 52. Analogical rules based on the rule from Figure 51 63](#_Toc137205246)

[Figure 53. Building obtained with the use of analogical rules 64](#_Toc137205247)

[Figure 54. Reverted rule based on the rule from Figure 51 64](#_Toc137205248)

[Figure 55. Buildings obtained with the use of a reverted rule 65](#_Toc137205249)

[Figure 56. A reference building with a pattern 65](#_Toc137205250)

[Figure 57. A grammar rule of attaching a pattern 66](#_Toc137205251)

[Figure 58. Buildings generated with the use of the rule from Figure 57 66](#_Toc137205252)

[Figure 59. Buildings generated with the use of the grammar without patterns 67](#_Toc137205253)

[Figure 60. A reference building made from seven components 67](#_Toc137205254)

[Figure 61. Derivative buildings made from several components 68](#_Toc137205255)

[Figure 62. Derivative buildings made from many of components 69](#_Toc137205256)

[Figure 63. Two architectural forms used as reference buildings 70](#_Toc137205257)

[Figure 64. Objects generated on the basis of two reference buildings 70](#_Toc137205258)

[Figure 65. Building components arranged in two rows 72](#_Toc137205259)

[Figure 66. Building components arranged in one row 72](#_Toc137205260)

[Figure 67. Partial mirror symmetry of a building 73](#_Toc137205261)

[Figure 68. Rotational symmetry of a building 74](#_Toc137205262)

[Figure 69. A building lacking stability 75](#_Toc137205263)

[Figure 70. A building in equilibrium 75](#_Toc137205264)

[Figure 71. A building made from three components 76](#_Toc137205265)

[Figure 72. A building made from thirty components 76](#_Toc137205266)

[Figure 73. A building containing three types of components 77](#_Toc137205267)

[Figure 74. A building containing one type of components 77](#_Toc137205268)

[Figure 75. Reference building 78](#_Toc137205269)

[Figure 76. Building derived from the building in Figure 75 () 78](#_Toc137205270)

[Figure 77. Building derived from the building in Figure 75 () 79](#_Toc137205271)

[Figure 78. Phenotype of a building containing two components 81](#_Toc137205272)

[Figure 79. Reference buildings 82](#_Toc137205273)

[Figure 80. Graph representations of the buildings from Figure 79 82](#_Toc137205274)

[Figure 81. Components grouped into a pattern 83](#_Toc137205275)

[Figure 82. Graph with a hierarchical node representing a pattern 84](#_Toc137205276)

[Figure 83. Bottom components of the reference buildings 85](#_Toc137205277)

[Figure 84. The axioms of the graph grammar based on the reference buildings from Figure 83 85](#_Toc137205278)

[Figure 85. Graph grammar created on the basis of the CP-graphs from Figure 80 87](#_Toc137205279)

[Figure 86. Analogical rules for the grammar in Figure 85 88](#_Toc137205280)

[Figure 87. Reverted rules for the grammar in Figure 85 89](#_Toc137205281)

[Figure 88. Graph grammar created on the basis of the CP-graphs from Figure 82 90](#_Toc137205282)

[Figure 89. Example of an initial population based on the grammar in Figure 85 91](#_Toc137205283)

[Figure 90. Example of an initial population based on the grammar in Figure 88 92](#_Toc137205284)

[Figure 91. Reference building with a pattern of stairs 92](#_Toc137205285)

[Figure 92. Initial population based on the building from Figure 91 93](#_Toc137205286)

[Figure 93. Reference building with a pattern of double stairs and a plinth 93](#_Toc137205287)

[Figure 94. Initial population based on the building from Figure 93 94](#_Toc137205288)

[Figure 95. Example of a phenotype (1) 94](#_Toc137205289)

[Figure 96. Phenotype obtained with a sequence of graph grammar rules 95](#_Toc137205290)

[Figure 97. Example of a phenotype (2) 96](#_Toc137205291)

[Figure 98. Phenotype of the first child 99](#_Toc137205292)

[Figure 99. Phenotype of the second child 100](#_Toc137205293)

[Figure 100. Mutation by altering metric attributes’ values – the resultant phenotype 100](#_Toc137205294)

[Figure 101. Mutation by removing a rule – the resultant phenotype 101](#_Toc137205295)

[Figure 102. Mutation by adding a rule – the resultant phenotype 102](#_Toc137205296)

[Figure 103. Reference building with high values of alignment and mirror symmetry 103](#_Toc137205297)

[Figure 104. Evaluation of the building from Figure 103 104](#_Toc137205298)

[Figure 105. Buildings from the initial population generated on the basis of the building in Figure 103 104](#_Toc137205299)

[Figure 106. Evaluation of the buildings from Figure 106 104](#_Toc137205300)

[Figure 107. Evolution results for the reference building showed in Figure 103 105](#_Toc137205301)

[Figure 108. Evaluation of the buildings from Figure 107 105](#_Toc137205302)

[Figure 109. Another initial population generated on the basis of the building in Figure 103 (with the use of graph grammar reverted rules) 105](#_Toc137205303)

[Figure 110. Evaluation of the buildings in Figure 109 106](#_Toc137205304)

[Figure 111. Evolution results for the reference building shown in Figure 103 (with the use of graph grammar reverted rules) 106](#_Toc137205305)

[Figure 112. Evaluation of the buildings in Figure 111 106](#_Toc137205306)

[Figure 113. Reference building with high value of mirror symmetry 107](#_Toc137205307)

[Figure 114. Initial population based on the building from Figure 113 107](#_Toc137205308)

[Figure 115. Evolution results for the reference building from Figure 113 108](#_Toc137205309)

[Figure 116. Two reference buildings 108](#_Toc137205310)

[Figure 117. Initial population based on the buildings from Figure 116 109](#_Toc137205311)

[Figure 118. Evolution results for the reference buildings from Figure 116 109](#_Toc137205312)

[Figure 119. Reference building with high values of both rotational and mirror symmetry 110](#_Toc137205313)

[Figure 120. Initial population based on the building from Figure 119 110](#_Toc137205314)

[Figure 121. Evolution results for the reference building from Figure 119 111](#_Toc137205315)

[Figure 122. Components attached to each other on the different sides of the same plane 114](#_Toc137205316)

[Figure 123. Components aligned to the same plane 114](#_Toc137205317)

[Figure 124. Lack of rotational symmetry in polygon groups 118](#_Toc137205318)

[Figure 125. Rotational symmetry in polygon groups 118](#_Toc137205319)

# Bibliography

1. Birkhoff, G.D. *Aesthetic Measure*; Cambridge Massachusetts Harvard University Press, 1933;

2. Biederman, I. Recognition-by-Components: A Theory of Human Image Understanding. In *Human Perception: Institutional Performance and Reform in Australia*; 2018.

3. Grabska, E. Graph and Designing. In Proceedings of the International Workshop on Graph Transformations in Computer Science; Schneider, H.J., Ehrig, H., Eds.; Springer: Berlin/Heidelberg, January 4 1993; pp. 188–202.

4. Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; 1992; Vol. 33;.

5. Mars, A. Automatic Generation of Architecture in Context. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); 2018; Vol. 11151 LNCS.

6. Mars, A.; Grabska, E. Towards an Implementable Aesthetic Measure for Collaborative Architecture Design. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); 2015; Vol. 9320.

7. Mars, A.; Grabska, E. Generation of 3D Architectural Objects with the Use of an Aesthetic Oriented Multi-Agent System. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); 2016; Vol. 9929 LNCS.

8. Mars, A.; Grabska, E.; Ślusarczyk, G.; Strug, B. Style-Oriented Evolutionary Design of Architectural Forms Directed by Aesthetic Measure. In *Design Computing and Cognition ’18*; 2019.

9. Mars, A.; Grabska, E.; Ślusarczyk, G.; Strug, B. Graph-Based Knowledge Inference for Style-Directed Architectural Design. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); 2019; Vol. 11734 LNAI.

10. Mars, A.; Grabska, E.; Ślusarczyk, G.; Strug, B. Design Characteristics and Aesthetics in Evolutionary Design of Architectural Forms Directed by Fuzzy Evaluation. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* **2020**, *34*, doi:10.1017/S0890060420000153.

11. Knight, T.W. Transformations in Design: A Formal Approach to Stylistic Change and Innovation in the Visual Arts. *Choice Reviews Online* **1995**, *32*, doi:10.5860/choice.32-6041.

12. M De Jong, T. Ways to Study and Research: Urban, Architectural and Technical Design 2019.

13. Marr, D. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. *Vision: a computational investigation into the human representation and processing of visual information.* **1982**, doi:10.1016/0022-2496(83)90030-5.

14. Floorplanner Available online: https://floorplanner.com/ (accessed on 5 July 2022).

15. Tarr, M.J.; Quoc, C.V. Visual Object Recognition 2002.

16. Bülthoff, H.H.; Edelman, S. Psychophysical Support for a Two-Dimensional View Interpolation Theory of Object Recognition. *Proc Natl Acad Sci U S A* **1992**, *89*, doi:10.1073/pnas.89.1.60.

17. Tarr, M.J. Rotating Objects to Recognize Them: A Case Study on the Role of Viewpoint Dependency in the Recognition of Three-Dimensional Objects. *Psychon Bull Rev* **1995**, *2*, doi:10.3758/BF03214412.

18. Perrett, D.I.; Oram, M.W.; Ashbridge, E. Evidence Accumulation in Cell Populations Responsive to Faces: An Account of Generalisation of Recognition without Mental Transformations. *Cognition* **1998**, *67*, doi:10.1016/S0010-0277(98)00015-8.

19. Riesenhuber, M.; Poggio, T. Hierarchical Models of Object Recognition in Cortex. *Nat Neurosci* **1999**, *2*, doi:10.1038/14819.

20. Marr, D.; Nishihara, H.K. Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. *Proceedings of the Royal Society of London. Series B, Containing papers of a Biological character. Royal Society (Great Britain)* **1978**, *200*, doi:10.1098/rspb.1978.0020.

21. Patel, L.N.; O’Brian Holt, P. SymGeons: A Prototype Model of Visual Complexity with Implications For Visual Control Tasks. *Cognition, Technology & Work* **2001**, *3*, doi:10.1007/pl00011532.

22. Slyadnev, S.; Malyshev, A.; Voevodin, A.; Turlapov, V. On the Role of Graph Theory Apparatus in a Cad Modeling Kernel. In Proceedings of the CEUR Workshop Proceedings; 2020; Vol. 2744.

23. Strug, B.; Paszyńska, A.; Paszyński, M.; Grabska, E. Using a Graph Grammar System in the Finite Element Method. *International Journal of Applied Mathematics and Computer Science* **2013**, *23*, doi:10.2478/amcs-2013-0063.

24. Singh, V.; Gu, N. Towards an Integrated Generative Design Framework. *Des Stud* **2012**, *33*, doi:10.1016/j.destud.2011.06.001.

25. Stephen, W. Statistical-Mechanics-Cellular-Automata-Stephen-Wolfram-Article.Pdf. *Rev. Mod. Phys.* 1983, *55*.

26. Neumann, J. General and Logical Theory of Automata. In Proceedings of the Cerebral Mechanisms in Behavior: The Hixon Symposium; 1951; Vol. 6.

27. Herr, C.M.; Kvan, T. Using Cellular Automata to Generate High-Density Building Form. In Proceedings of the Computer Aided Architectural Design Futures 2005 - Proceedings of the 11th International CAAD Futures Conference; 2005.

28. Krawczyk, R.J. Architectural Interpretation of Cellular Automata. *Proceedings of the Generative Art Conference* **2002**.

29. Çalışır Adem, P.; Çağdaş, G. Computational Design Thinking through Cellular Automata: Reflections from Design Studios. *Journal of Design Studio* **2020**, doi:10.46474/jds.816833.

30. Frazer, john Computing without Computers. *Architectural Design* **2005**, *75*, 34–43.

31. Frazer, J. An Evolutionary Architecture. *Architectural Association publications* **1995**, *Themes VII*.

32. Herr, C.M.; Kvan, T. Adapting Cellular Automata to Support the Architectural Design Process. *Autom Constr* **2007**, *16*, doi:10.1016/j.autcon.2005.10.005.

33. Herr, C.M. From Form Generators to Automated Diagrams: Using Cellular Automata to Support Architectural Design. *Department of Architecture* **2008**, *Doctor of*.

34. Buelow, P. v.; Falk, A.; Turrin, M. Optimization of Structural Form Using a Genetic Algorithm to Search Associative Parametric Geometry. In Proceedings of the Structures and Architecture - Proceedings of the 1st International Conference on Structures and Architecture, ICSA 2010; 2010.

35. Salajegheh, E.; Mashayekhi, M.; Khatibinia, M.; Kaykha, M. Optimum Shape Design of Space Structures by Genetic Algorithm. *International Journal of Space Structures* **2009**, *24*, doi:10.1260/026635109788251502.

36. Lindenmayer, A. Mathematical Models for Cellular Interactions in Development II. Simple and Branching Filaments with Two-Sided Inputs. *J Theor Biol* **1968**, *18*, doi:10.1016/0022-5193(68)90080-5.

37. Kari, L.; Rozenberg, G.; Salomaa, A. L SYSTEMS. **1974**, doi:10.1007/978-3-642-59136-5\_5.

38. Prusinkiewicz, P.; Lindenmayer, A. *The Algorithmic Beauty of Plants*; 1990;

39. Michael Hansmeyer - Computational Architecture Available online: https://www.michael-hansmeyer.com/ (accessed on 5 July 2022).

40. Beni, G.; Wang, J. Swarm Intelligence in Cellular Robotic Systems. In *Robots and Biological Systems: Towards a New Bionics?*; 1993.

41. Chen, Y. Yuxing Chen Available online: https://www.yuxingc.com/ (accessed on 5 July 2022).

42. Miranda, P.; Coates, P. Swarm Modelling. The Use of Swarm Intelligence to Generate Architectural Form. *3th International Conference on Generative Art* **2000**.

43. Kartal, H.B.; Kartal, A.N. SWARM INTELLIGENCE IN ARCHITECTURAL DESIGN AS A TECHNIQUE OF AI. *ArtGRID - Journal of  Architecture Engineering and Fine Arts* **2020**, *2*, 122–129.

44. Alisa Andrasek’s Algorithmic Seroussi Pavilion Available online: https://www.evolo.us/alisa-andraseks-algorithmic-seroussi-pavilion/ (accessed on 5 July 2022).

45. Quinze, A. The Sequence Available online: https://www.arnequinze.com/art-and-exhibitions/the-sequence (accessed on 5 July 2022).

46. McDermott, J. Graph Grammars for Evolutionary 3D Design. *Genet Program Evolvable Mach* **2013**, *14*, doi:10.1007/s10710-013-9190-0.

47. Stiny, George.; Gips, J. *Algorithmic Aesthetics, Computer Models for Criticism and Design in the Arts*; 1978;

48. Liew, H. Extending Shape Grammars with Descriptors. In *Design Computing and Cognition ’04*; 2004.

49. Stiny, G.; Mitchell, W.J. The Palladian Grammar. *Environ Plann B Plann Des* **1978**, *5*, doi:10.1068/b050005.

50. Buelinckx, H. Wren’s Language of City Church Designs: A Formal Generative Classification. *Environ Plann B Plann Des* **1993**, *20*, doi:10.1068/b200645.

51. Downing, F.; Flemming, U. The Bungalows of Buffalo. *Environ Plann B Plann Des* **1981**, *8*, doi:10.1068/b080269.

52. Esri CityEngine Available online: https://www.esri.pl/produkty/esri-cityengine/ (accessed on 29 May 2023).

53. Rozenberg, G. *Handbook of Graph Grammars and Computing by Graph Transformations*; World Scientific Publishing, 1997; Vol. 1–3;.

54. Grabska, E.; Strug, B.; Ślusarczyk, G. Educational Facility Design with Graph Grammar Systems. In Proceedings of the CEUR Workshop Proceedings; 2019; Vol. 2394.

55. Du, X.; Jiao, J.; Tseng, M.M. Graph Grammar Based Product Family Modeling. *Concurr Eng Res Appl* **2002**, *10*, doi:10.1177/1063293X02010002635.

56. Tadeusiewicz, R. *Sieci Neuronowe*; Akademicka Oficyna Wydawnicza: Warszawa, 1993;

57. Radziszewski, K. Artificial Neural Networks as an Architectural Design Tool-Generating New Detail Forms Based on the Roman Corinthian Order Capital. In Proceedings of the IOP Conference Series: Materials Science and Engineering; 2017; Vol. 245.

58. Midjourney Available online: https://www.midjourney.com (accessed on 29 May 2023).

59. DALL-E 2 Available online: https://openai.com/product/dall-e-2 (accessed on 29 May 2023).

60. Saunders, R.; Gero, J.S. Curious Agents and Situated Design Evaluations. In Proceedings of the Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM; 2004; Vol. 18.

61. Soddu, C. 20 Years ARGENIA Evolution 2018.

62. Pinel, F.; Varshney, L.R.; Bhattacharjya, D. *Computational Creativity Research: Towards Creative Machines*; 2015; Vol. 7;.

63. Boden, M.A. Creativity as a Neuroscientific Mystery. In *Neuroscience of Creativity*; 2013.

64. Gero, J.S. Design Prototypes. A Knowledge Representation Schema for Design. *AI Mag* **1990**, *11*.

65. Gero, J.S. Creativity, Emergence and Evolution in Design. *Knowl Based Syst* **1996**, *9*, doi:10.1016/S0950-7051(96)01054-4.

66. Coyne, R.D.; Gero, J.S. Design Knowledge and Context. *Environ Plann B Plann Des* **1985**, *12*, doi:10.1068/b120419.

67. Krippendorff, K. The Semantic Turn: A New Foundation for Design. *Artifact* **2007**, *1*, doi:10.1080/17493460600844157.

68. Gero, J.S.; Louis, S.J.; Kundu, S. Evolutionary Learning of Novel Grammars for Design Improvement. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing* **1994**, *8*, doi:10.1017/S089006040000069X.

69. Strug, B. Zastosowanie Algorytmów Ewolucyjnych i Transformacji Grafowych w Projektowaniu Graficznym Wspomaganym Komputerowo, IPPT: Warsaw, 2002.

70. Hliniak, G.; Strug, B. Graph Grammars and Evolutionary Methods in Graphical Design. *Machine Graphic s and Vision* **2000**, *9*.

71. Kawano, H. What Is Computer Art? In *Artist and Computer*; Leavitt, R., Ed.; Creative Computing Press: Morristown, 1976; pp. 112–113.

72. Greenfield, G.; MacHado, P. Guest Editors’ Introduction. *Journal of Mathematics and the Arts* 2012, *6*.

73. Hoenig, F. Defining Computational Aesthetics. *Computational Aesthetics’05 Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging* **2005**.

74. Machado, P.; Cardoso, A. LNAI 1515 - Computing Aesthetics. In Proceedings of the Advances in Artificial Intelligence; 1998.

75. Brachmann, A.; Redies, C. Computational and Experimental Approaches to Visual Aesthetics. *Front Comput Neurosci* 2017, *11*.

76. Galanter, P. Computational Aesthetic Evaluation: Past and Future. In *Computers and Creativity*; 2012; Vol. 9783642317279.

77. Friedenberg, J.; Bertamini, M. Aesthetic Preference for Polygon Shape. *Empirical Studies of the Arts* **2015**, *33*, doi:10.1177/0276237415594708.

78. Staudek, T. On Birkhoff’s Aesthetic Measure of Vases 1999.

79. Tarko, J.; Grabska, E. Aesthetic Measure for Three-Dimensional Objects. *Machine Graphics and Vision* **2011**, *20*.

80. Stufano Melone, M.R. Towards an Ontology for Measure of Music Aesthetics. In Proceedings of the CEUR Workshop Proceedings; 2019; Vol. 2518.

81. Megahed, Y.S.; Gabr, H.S. Quantitative Methods of Architectural Aesthetic Assessment. *International Association of Empirical Aesthetics Congress* **2008**.

82. Kalonaris, S.; Jordanous, A. Computational Music Aesthetics: A Survey and Some Thoughts. In Proceedings of the 3rd Conference on Computational Simulation of Musical Creativity; 2018.

83. Florence, P.S.; Zipf, G.K. Human Behaviour and the Principle of Least Effort. *The Economic Journal* **1950**, *60*, doi:10.2307/2226729.

84. Manaris, B.; Vaughan, D.; Wagner, C.; Romero, J.; Davis, R.B. Evolutionary Music and the Zipf-Mandelbrot Law: Developing Fitness Functions for Pleasant Music. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2003**, *2611*, doi:10.1007/3-540-36605-9\_48.

85. Manaris, B.; Romero, J.; Machado, P.; Krehbiel, D.; Hirzel, T.; Pharr, W.; Davis, R.B. Zipf’s Law, Music Classification, and Aesthetics. *Computer Music Journal* **2005**, *29*, doi:10.1162/comj.2005.29.1.55.

86. Emmery, C. The Aesthetic Value of Zipfan Vsual Art 2013.

87. Tong, H.; Li, M.; Zhang, H.J.; He, J.; Zhang, C. Classification of Digital Photos Taken by Photographers or Home Users. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2004**, *3331*, doi:10.1007/978-3-540-30541-5\_25.

88. Ke, Y.; Tang, X.; Jing, F. The Design of High-Level Features for Photo Quality Assessment. In Proceedings of the Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition; 2006; Vol. 1.

89. Datta, R.; Wang, J.Z. ACQUINE.; 2010.

90. Lu, X.; Lin, Z.; Jin, H.; Yang, J.; Wang, J.Z. Rating Image Aesthetics Using Deep Learning. *IEEE Trans Multimedia* **2015**, *17*, doi:10.1109/TMM.2015.2477040.

91. Marchesotti, L.; Perronnin, F.; Larlus, D.; Csurka, G. Assessing the Aesthetic Quality of Photographs Using Generic Image Descriptors. In Proceedings of the Proceedings of the IEEE International Conference on Computer Vision; 2011.

92. Lowe, D.G. Object Recognition from Local Scale-Invariant Features. In Proceedings of the Proceedings of the IEEE International Conference on Computer Vision; 1999; Vol. 2.

93. Spratt, E.L.; Elgammal, A. Computational Beauty: Aesthetic Judgment at the Intersection of Art and Science. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); 2015; Vol. 8925.

94. Shamir, L. What Makes a Pollock Pollock: A Machine Vision Approach. *International Journal of Arts and Technology* **2015**, *8*, doi:10.1504/IJART.2015.067389.

95. Condorovici, R.; Florea, C.; Vertan, C. Author Identification for Digitized Paintings Collections. In Proceedings of the ISSCS 2013 - International Symposium on Signals, Circuits and Systems; 2013.

96. Tan, W.R.; Chan, C.S.; Aguirre, H.E.; Tanaka, K. Ceci n’est Pas Une Pipe: A Deep Convolutional Network for Fine-Art Paintings Classification. In Proceedings of the Proceedings - International Conference on Image Processing, ICIP; 2016; Vol. 2016-August.

97. Saleh, B.; Abe, K.; Arora, R.S.; Elgammal, A. Toward Automated Discovery of Artistic Influence. *Multimed Tools Appl* **2016**, *75*, doi:10.1007/s11042-014-2193-x.

98. Li, J.; Yao, L.; Hendriks, E.; Wang, J.Z. Rhythmic Brushstrokes Distinguish van Gogh from His Contemporaries: Findings via Automated Brushstroke Extraction. *IEEE Trans Pattern Anal Mach Intell* **2012**, *34*, doi:10.1109/TPAMI.2011.203.

99. Li, C.; Chen, T. Aesthetic Visual Quality Assessment of Paintings. *IEEE Journal on Selected Topics in Signal Processing* **2009**, *3*, doi:10.1109/JSTSP.2009.2015077.

100. Saleh, B.; Elgammal, A. Large-Scale Classification of Fine-Art Paintings: Learning the Right Metric on the Right Feature. *arXiv:1505.00855* 2015.

101. Arora, R.S.; Elgammal, A. [25] Arora, R. S., and Elgammal, A. (2012). “Towards Automated Classification of Fineart Painting Style: Comparative Study. In Proceedings of the 2012 21st International Conference on the Pattern Recognition (ICPR) Tsukuba: IEEE; 2012; pp. 3541–3544.

102. Berg, M. Phi, the Golden Ratio (to 4599 Decimal Places) and Fibonacci Numbers. *The Fibonacci Quarterly* **1961**, *4*, 157–162.

103. Wu, Y.-S. Aesthetic Preference in the Composition Ratios of Graphic Designs. In Proceedings of the Engineering Psychology and Cognitive Ergonomics: 19th International Conference, EPCE 2022, Held as Part of the 24th HCI International Conference, HCII 2022; 2022; pp. 137–145.

104. Liu, Y.; Zhang, Q. Interface Design Aesthetics of Interaction Design. In Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); 2019; Vol. 11583 LNCS.

105. Mayatskaya, I.; Yazyev, B.; Yazyeva, S. Harmony and Disharmony as the Architectural Appearance Features of the City. *E3S Web of Conferences* **2021**, *281*, doi:10.1051/e3sconf/202128102004.

106. Willa „Savoye” Available online: https://pl.wikipedia.org/wiki/Willa\_%E2%80%9ESavoye%E2%80%9D (accessed on 5 July 2022).

107. Nekosuki Hōryū-Ji Available online: https://en.wikipedia.org/wiki/H%C5%8Dry%C5%AB-ji (accessed on 5 July 2022).

108. Falconer, K. *Fractal Geometry*; Wiley, 2003;

109. Norton, A.; Edgar, G.A. Measure, Topology, and Fractal Geometry. *The American Mathematical Monthly* **1992**, *99*, doi:10.2307/2324919.

110. Spehar, B.; Clifford, C.W.G.; Newell, B.R.; Taylor, R.P. Universal Aesthetic of Fractals. *Computers and Graphics (Pergamon)* **2003**, *27*, doi:10.1016/S0097-8493(03)00154-7.

111. Eglash, R. African Fractals Available online: https://span.uncg.edu/fractals/ (accessed on 5 July 2022).

112. Eglash, R. Fractals in African Settlement Architecture. *Complexity* **1998**, *4*, doi:10.1002/(SICI)1099-0526(199811/12)4:2<21::AID-CPLX6>3.0.CO;2-F.

113. Wertheimer, M. Experimental Studies on the Seeing of Motion. *Z Psychol* **1912**, *61*, 161–265.

114. Wagemans, J.; Elder, J.H.; Kubovy, M.; Palmer, S.E.; Peterson, M.A.; Singh, M.; von der Heydt, R. A Century of Gestalt Psychology in Visual Perception: I. Perceptual Grouping and Figure-Ground Organization. *Psychol Bull* **2012**, *138*, doi:10.1037/a0029333.

115. Gropius, W. *Scope of Total Architecture*; Harper: New York, 1955;

116. Borowik, A. Teoria Formy Architektonicznej Juliusza Żórawskiego i Jego Uczniów Na Tle Koncepcji Psychologii Postaci Wypracowanych w Bauhausie. *Sztuka i Dokumentacja* **2020**, *23*, 121–131.

117. Formative Years. In *Bauhaus and Bauhaus People*; Neumann, E., Ed.; Van Nostrand Reinhold: New York, 1993; pp. 209–209.

118. Behrens, R. Gestalt Theory and Bauhaus - A Correspondence. *Gestalt Theory* **2012**, *34*, 87–87.

119. Zórawski, J. *O Budowie Formy Architektonicznej*; Arkady: Warszawa, 1962;

120. Grycuk, A. Juliusz Żórawski (Architekt) Available online: https://pl.wikipedia.org/wiki/Juliusz\_%C5%BB%C3%B3rawski\_(architekt) (accessed on 9 July 2022).

121. Gardner, B.; Krishnamurti, R. Gardner, Brian; Krishnamurti, Ramesh (2018): Ordering the Aesthetic (A+) in Architecture: Advancing a Theory of Modular Computation 2018.

122. Vinchu, Ar.G.N.; Jirge, N.; Deshpande, Ar.A. Application of Aesthetics in Architecture and Design. *International Journal of Engineering Research and Technology.* **2017**, *10*.

123. Glattfelder, J.B. The Semantics of Symmetry, Invariance, and Structure. In; 2019.

124. Machilsen, B.; Pauwels, M.; Wagemans, J. The Role of Vertical Mirror Symmetry in Visualshape Detection. *J Vis* **2009**, *9*, doi:10.1167/9.12.11.

125. Brass, P. On Finding Maximum-Cardinality Symmetric Subsets. In Proceedings of the Computational Geometry: Theory and Applications; 2003; Vol. 24.