Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

INS

Information Systems

INformation Systems

Generation of abstract geometric art based on exact aesthetics, gestalt theory and graphic design principles

M.W. Kauw-A-Tjoe

REPORT INS-E0509 APRIL 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3681

Generation of abstract geometric art based on exact aesthetics, gestalt theory and graphic design principles

ABSTRACT
In this thesis artificial intelligence ideas are applied to the domain of fine arts and especially modern art. First, we take a closer look at avant garde art movements of the late 19th and first half of the 20th century. After that, we make an analysis of the knowledge on which this art movement is partly based by considering the fields of aesthetics, gestalt psychology and graphic design. Having formalised general ideas about what a well-formed painting should consist of, we then look at ways of incorporate these ideas in a model for generating a composition. We design a formal framework to which we map the domain concepts. Based on the framework, we make a top-down knowledge decomposition. To demonstrate how our ideas can be applied in a practical situation, we have implemented a prototype system. Theoretically, this system is split up into the front-end part, in which the actual output is generated, and the back-end part, in which artificial intelligence techniques are applied to the actual concepts of composing an artwork. The front-end is partly based on the multimedia generation system called Cuypers, which was developed at the Centrum voor Wiskunde en Informatica (CWI) in Amsterdam. Cuypers was made to generate dynamic presentations and therefore generates XML, which subsequently is transformed into a desired format (XHTML, TIME, SMIL) using XSLT transformation stylesheets. Our system generates output in the Scalable Vector Graphics (SVG) format, which is an XML based standard for vector graphics and animation. The back-end part is based on the formal ideas about art described above. It is implemented in Eclipse Prolog. Finally, we discuss the artistic significance of our results. We conclude this thesis by discussing the advantages and disadvantages of the conceptual decisions, as well as suggesting directions for future research, including ideas for the evaluation of the generated compositions.

2000 Mathematics Subject Classification: See ACM
1998 ACM Computing Classification System: D.1.6, H.5.1, I.3.5, J.5
Keywords and Phrases: artificial intelligence; generative art; shape grammars
Note: Masters thesis for Vrije Universiteit, Amsterdam

Generation of Abstract Geometric Art Based on Exact Aesthetics, Gestalt Theory and Graphic Design Principles.

M.W. Kauw-A-Tjoe
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT

In this thesis artificial intelligence ideas are applied to the domain of fine arts and especially modern art. First, we take a closer look at avant garde art movements of the late 19th and first half of the 20th century. After that, we make an analysis of the knowledge on which this art movement is partly based by considering the fields of aesthetics, gestalt psychology and graphic design.

Having formalised general ideas about what a 'well-formed' painting should consist of, we then look at ways of incorporate these ideas in a model for generating a composition. We design a formal framework to which we map the domain concepts. Based on the framework, we make a top-down knowledge decomposition.

To demonstrate how our ideas can be applied in a practical situation, we have implemented a prototype system. Theoretically, this system is split up into the front-end part, in which the actual output is generated, and the back-end part, in which artificial intelligence techniques are applied to the actual concepts of composing an artwork.

The front-end is partly based on the multimedia generation system called *Cuypers*, which was developed at the *Centrum voor Wiskunde en Informatica* (CWI) in Amsterdam. Cuypers was made to generate dynamic presentations and therefore generates XML, which subsequently is transformed into a desired format (XHTML, TIME, SMIL) using XSLT transformation stylesheets. Our system generates output in the Scalable Vector Graphics (SVG) format, which is an XML based standard for vector graphics and animation. The back-end part is based on the formal ideas about art described above. It is implemented in Eclipse Prolog.

Finally, we discuss the artistic significance of our results. We conclude this thesis by discussing the advantages and disadvantages of the conceptual decisions, as well as suggesting directions for future research, including ideas for the evaluation of the generated compositions.

1998 ACM Computing Classification System: D.1.6, H.5.1, I.3.5, J.5
Keywords and Phrases: artificial intelligence, generative art, shape grammars

Table of Contents

Chapter 1
Introduction

The domain of this project is that of *generative art* in a computer science context: 'art that is generated by a computer(program)'. In our situation, the domain of generative art could roughly be split up into three sub-domains. First, a division can be made between art and science. These fields are fairly opposite to each other, a fact which will turn out to raise several fundamental questions. Furthermore, a useful distinction can be made in the domain of computer science between the fields of artificial intelligence and web technology, both of which will play a significant role in the implementation of our ideas.

We are mainly interested in abstract painting, so we will take a close look at the fine arts and especially the avantgarde art movements of the late 19th and first half of the 20th century. These movements are strongly linked with some theoretical disciplines within philosophy, psychology and the design community, which makes them suitable for this kind of research.

The research of this project is done for a masters degree in artificial intelligence (AI). Therefore, after we have made an in-depth analysis of the knowledge domain, we will derive formal ideas on which the back-end of the system is based. These ideas include generative concepts that provide a basis to build an artwork-model from scratch. Also, we will discuss ways to evaluate the generated artwork from a more holistic point of view.

Whereas AI fulfills the need to create a conceptually sound research design and art theory provides a solid basis for this, existing web technology can provide a framework for the generative process. We have chosen Scalable Vector Graphics (SVG), the XML standard for vector graphics, as the format to visualise the concepts that are presented in this thesis.

There exist other projects in which AI techniques are combined with art. Cohen for example, who wrote some interesting philosophically oriented papers on the possibility of machines being creative (for example [16]) implemented a system called AARON, which is able to create paintings in a certain style. The essential difference

with this project is the fact that Cohen is more concerned with the consequences of AI technology for artistic content (he states that the compositions that AARON produces are not art), while we take a more utilitarian approach by researching the possibilities of applying AI within a specific art domain.

Our research question is the following: 'How can artificial intelligence techniques be combined with vector graphics to create geometric abstract art?'

Our approach is to formalise knowledge derived from art movements and related scientific disciplines such as gestalt psychology and aesthetics. Based on this knowledge we will build a system that implements some of these ideas and generates output in Scalable Vector Graphics (SVG).

Certain art movements, such as dadaism and post-modernism, that take a more nihilistic or eclectic approach, are beyond the scope of this thesis, because their principles are focused on content rather than on form. Therefore they are difficult to formalise. Also, to reduce the complexity of the problem space, we will restrict ourselves to two-dimensional compositions while ignoring interesting technical possibilities such as animation and audio. Depth composition will consist of placing shapes in layers on top of each other in the two-dimensional plane of the composition area. This is inherent to our choice for vector graphics.

The structure of this thesis is the following:

Chapter 2 explains the theoretical background of our three primary domains: modern art, artificial intelligence and web technology.

Chapter 3 translates theory to formal knowledge. We design a conceptual framework to which the concepts discussed in chapter 2 are mapped.

Chapter 4 discusses the implementation of our demo and the extent to which we can generate art.

Chapter 5 Concludes the thesis by analyzing the main conceptual decisions that have been made and discusses possibilities for future research.

Chapter 2
Theoretical Background

In this chapter, relevant background information with respect to generative art is given. In the first section, we take a look at the creative issues involved. After that, we look at some possible artificial intelligence approaches. In the third section, we discuss available web technology for the representation and generation of two-dimensional graphics.

1. Art Perspective

The domain of the fine arts is virtually without bounds and therefore choices have to be made to limit our field of interest. Because this is a computer science thesis, we are looking for an art movement of which we are able to formalise the principles to some extent. Besides that, it should be possible to generate output that resembles compositions in that movement's style at a certain level. As will become clear soon, abstract geometric art fits these requirements. Therefore, in this section, the domain of modern arts is explored. First, we take a look at the history of modern art and how early twentieth century geometric painting movements have evolved. After that, we look at existing combinations of art and computer technology. We discuss some general issues concerning the gap between technology and creativity while saving a more in-depth analysis for the discussion. In the last part of this section, a first step is taken to derive concrete formalisms out of which we can build an art-generation framework.

1.1 A Short History of Modern Art and Expressionism

Any contemporary painting could be classified as being modern. Historically speaking though, a modern painting is a painting that is based on ideas that were developed during the period of the modern art movement, which developed between 1860 and the second world war (to avoid confusion, modern art is often called *modernism*). Based on [25], we place abstract geometric art to the background of modern art in this section.

Art in the classic sense is mainly concerned with depicting reality, or nature, in an

accurate way. Well known 'classic' techniques are the use of proportional measures, perspective, foreshortening and chiaroscuro (the contrast between light and dark). Of course, artists always have maintained the freedom of expression to a certain degree. Whereas some paintings are hardly distinguishable from a picture, others depict a more abstract expression of the inner state of the artist. The first nineteenth century art movement in which expression played a significantly more important role is called *impressionism*. The impressionists took more liberties with respect to realism, mainly in the use of colour. Important impressionist painters such as Monet, Morisot, Renoir and Manet experimented with new compositions, often derived from *Ukiyo-e*, Japanese Edo-period wood prints. Although impressionistic painters changed general principles of form, thematically they were rather limited; many of them ignored the social developments of society. Instead, they stuck to bourgeois themes such as garden parties and idyllic landscapes. *Neo-impressionists* such as Pisarro, Degas and Van Gogh successfully incorporated more social themes into their paintings.

Paul Cézanne (1839-1906), a contemporary of the late-impressionists, took the idea of expression much further. He paid a lot of attention to the integrity of the painting itself, carefully arranging the composition at the level of brushstrokes. Thus, he not only gave a subjective view through the use of colour, but radically changed the aesthetic values and notions of art in general (a typical example of one of Cézanne's works is depicted in figure 2.1(a)). Yet his work was still based on the depiction of realistic objects.

Around the turn of the century, Cézannes work became widely appreciated and his ideas started to influence others. Throughout the first half of the twentieth century, many new movements appeared that emphasised expression through painting. Therefore, they are placed together under the term *expressionism*. Many expressionist painters started to experiment with *non-representational* or *non-figurative* abstractions. This eventually led to the abstract geometric art forms that are the basis for this thesis. Here follows a short summary of the most important expressionistic movements.

Paris, at that moment the cultural centre of the Western world, produced one of the first groups of expressionistic painters, including Matisse, Rousseau, Derain and De Vlaminck. They were highly influenced by African and other exotic art and at their first large exposition together they were named *'Les Fauves', the savages*. At the same time German painters also showed a large tendency towards expressionism.

The ideas fundamental to the movement know as *cubism* were originally developed by Picasso and Braque. Although they have always avoided an exact definition of cubism, they based their ideas on the simultaneously displaying of multiples sides of three-dimensional objects and tended to represent human forms and objects in terms of patterns and rhythms of geometric shapes. This 'primitive' or 'naive' way of expression was influenced by Cézanne as well as exotic cultures. Thus, a strong relationship can be seen between cubism and 'Les Fauves'. Figure 2.1(b) depicts the cubistic 'Woman in Armchair' by Picasso.

Futurism was founded in Italy in 1909 and was originally more concerned with poetry than with painting. The futurists wrote manifestos in which they embraced the industrialised world, which was dominated by technology and urbanisation. This is in strong contrast to previous movements such as impressionism, which emphasised the beauty and purity of nature. Futurism has produced only a few significant works, most of which were made by Boccionni. On the other hand, futurism's switch in mentality

has influenced practically every other Western art movement at the time.

With the threat of the first world war in the back of their minds, they portrayed scenes of social unrest, just like the neo-impressionists. One of the main differences was their emphasis on the emotional intensity and spontaneity of the situation. One of the first official expressionistic groups from Germany is called *'Die Brücke'*, with Kirchner as its best-known member.

Vassily Kandinsky, a Russian painter who moved to Germany, was the founder of another expressionistic group, *'Der Blaue Reiter'*. Kandinsky was very interested in science and gave up a professorship for art. Rudolf Steiner, one of the founders of gestalt psychology, attended his lectures in the *'Bauhaus'*, also known for the art movement of the same name. Kandinsky gradually worked towards new levels of expression, until he totally denounced representational forms and made compositions entirely based on geometric abstractions. A good example of one of these entirely geometric abstract paintings is depicted in figure 2.1(d).

In Russia, for example, paintings by important contemporary artists from Europe were imported by wealthy merchants. This way, Russian painters discovered cubism and futurism. By 1913, the first Russian expressionist art movements were started, eventually resulting in Kasimir Malevich's *suprematism* and Popova and Tatlin's *constructivism*. Figure 2.1(e) depicts a composition by Malevich, while figure 2.1(c) shows a canvas painted by Popova. The Russian painters placed the forms of expressionism in a much more social context, believing that the abstract ideas conveyed through their paintings had a quality of purity which corresponded to contemporary political ideas.

When the first world war broke out, many important painters were sent into the trenches, never to return. Some of them, including Kandinsky, managed to escape to Britain. After the war, abstract art was further developed by the *'Bauhaus'* movement in Germany and the *'De Stijl'* movement in the Netherlands, also known as *neo-plasticism*. Prominent artists of both movements, among others Gropius, Kandinsky, Mondrian, Rietveld and Van Doesburg, exchanged ideas with each other during lectures in the 'Bauhaus'-building. Many of them were influenced by earlier expressionist movements as well as by architects such as Lloyd-Wright and Berlage. Innovations were made in painting as well as architecture and industrial design. Geometric art reached its peak with the lattice compositions by Mondrian and the 'Schröder'-house by Rietveld. Figures 2.1(g) and 2.1(f) depict two typical neo-plasticist compositions.

The ideologies that underlay many late expressionist painting movements often turned out to be their weakness. The Russian expressionist art movements for example were banned because of their strong visions. Instead, painters had to return to traditional realism to depict common situations of the working class for propaganda purposes. Futurism is said to have a close relationship with fascism. Even the art of the neo-plasticists, which was allegedly based on spiritual ideas resulting from a calvinistic background, was limited by its strong ethical foundations.

After world war two, some artists still painted in the modernist tradition (for example the American abstract expressionists such as Jackson Pollock), but the bulk of new artists decided to take new directions. The content-based *post-modernist* approach to art is fundamentally opposite to that of modernism, which concentrates on structure and form. Still, modernism has had a great influence as one of the most important artistic changes in modern history. We continue the discussion about art in general in

chapter 5.

1.2 Computers & Art

The first computer artists appeared around 1950 [27]. Due to the formal nature of computers, strong emphasis lay on the mathematical approach of visuals. Most of the pioneers of computer art were active in the field of science. Franke, a German mathematician, did experiments with oscilloscopes of which he made photographic images. At the same time, in the USA Laposky independently produced work that was strikingly similar to that of Franke. Laposky was primarily an artist, although he started from mathematics. Thus, Franke and Laposky can be said to have similar interests but different priorities. Whitney Senior, on the other hand, had an artistic background and was most explicitly influenced by modernism. His aim was to create new, abstract films that were meant to represent music in a visual manner. He learned to work with computers to achieve this goal.

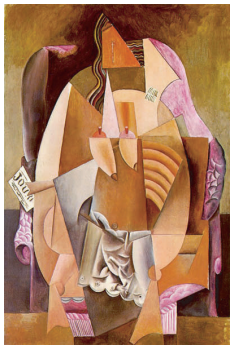
The use of geometry to create visuals resulted in the fact that early computer art was primarily concerned with form, just like much modern art. The sad thing is that once the technology to properly generate visuals with computers was available, the last significant modernist developments had been made twenty years earlier. The post-modern art that was fully in development at the time was primarily concerned with ideas about content and the implications of art itself. Ideas such as these are much more difficult to formalise.

This does not imply that no interesting work has been done in the field of computer arts. Many new developments have been driven by technology. For a couple of decades, generating graphics was too expensive and too inaccessible for regular artists. In the mid-eighties, personal computers and game consoles provided a wider public with graphics software. In [27], the year 1986 is denoted as a landmark because of three facts; first, the BBC broadcast a television series about artists who used a primitive graphics system that connected to a television set. Second, Andy Warhol, the wellknown *pop art* artist, made some works with a Commodore Amiga. Also in that year, the first version of Adobe's Photoshop, one of the most popular graphics tools to this date, was written.

Eventually, computer graphics have become available to everyone who has access to a computer. Popularisation and wide availability of common graphics applications has led to the fact that fancy graphics are now easily created with pre-programmed filters called *plug-ins*. On the other hand, many professional tools have been developed. Artists are able to explore the new boundaries of art in digital media.

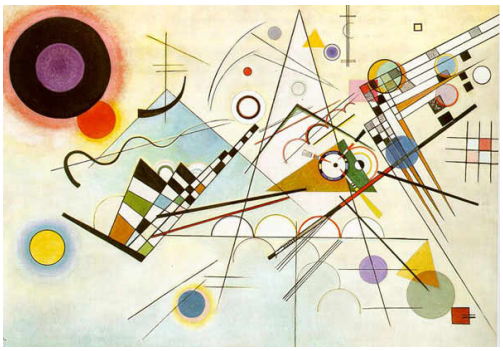
As for 'serious' computer art, there a couple of different approaches. The first is called *algorithmic art* and is based on computer algorithms that specify how an artwork should be built up. In this case, procedural computer algorithms are an intermediate through which the goal of the artist/programmer of expressing himself is reached. *Generative art* is to algorithmic art what AI is to computer science; ideas of creativity and more abstract, generational knowledge are implemented to give the system a level of autonomy. One way to achieve this is to make use of rule based reasoning, which is provided by fifth generation inferencing languages such as Prolog [8]. One could also using machine learning techniques to generate graphics. These include neural networks and evolutionary algorithms. Machine learning is not based on sets of rules but on a dynamic process. This way, patterns can emerge. More information about

(a) ‘Bend in Road’, Cézanne

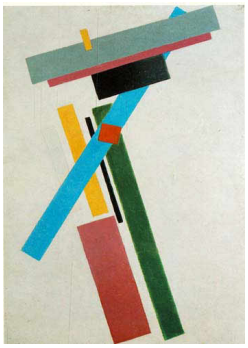


(b) ‘Woman in Armchair’, Picasso

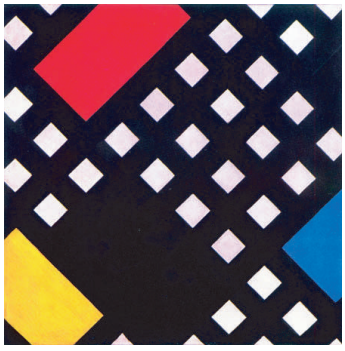
(c) ‘Painterly Architectonics’, Popova



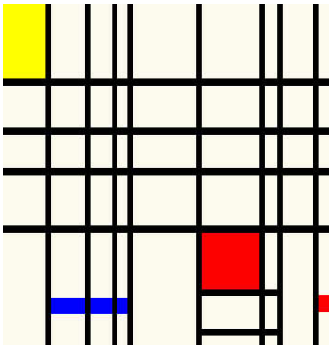
(d) ‘Composition 8’, Kandinsky



(e) ‘Petersburg’, Malevich



(f) ‘Countercomposition XV’, Van Doesburg



(g) ‘Composition with red, yellow and blue’, 1939-1942, Mondrian

Figure 2.1: Examples of abstract paintings

reasoning and machine learning in the context of graphics is given in section 2.

1.3 "But is it really art?"

One of the main philosophical questions concerning generative art is: "To what extent can something that is produced by a machine genuinely be called art?" Artificial creativity, in this light, could be seen as a form of intelligence. Therefore this question is similar to the classical question: "To what extent can a program be genuinely called intelligent?". This interesting issue is encountered in every field of artificial intelligence. Before referring to the discussion of our results in chapter 4, we shortly discuss of the distinctions between the so-called *strong* and the *weak AI claim* in the context of art [16, 20] here.

The weak claim states that human cognitive processes can be simulated by a sufficiently complex computer program. By contrast, the strong claim holds that not only the input and output of the computer are the same as a humans, but also the cognitive internal state of the machine is the same. Translating these notions to artificial creativity, the weak answer to the question at the start of the previous paragraph would be: "If the graphic output is similar to the art produced by a human artist, then it can be called art". Contrarily, the strong answer would be: "Besides generating qualitatively equal graphics, the machine must possess an internal state that represents the creative process."

Another point of view would be to regard the programmer of a piece of working generative-art software as the real artist. This is comparable to so-called *meta-artists* [27], who describe the way their work should be made, while leaving the exact instantiation of the details to others. We discuss higher level artistic considerations in more depth in chapter 4.

1.4 Domain Theory

In the previous sections we have discussed some historic and philosophical backgrounds of our domain. In this section, we handle the more theoretical, semi-formal side. We take a top-down approach; first we discuss the philosophy behind art and beauty, *aesthetics*, then we take a more practical, scientific approach by considering *gestalt psychology* and finally we look at how theory relates to common practice by taking a look at *composition* and *graphic design*.

Aesthetics The philosophical movement that considers the creation and evaluation of art is called aesthetics. A formal definition[1]: *'Aesthetics or æsthetics - The branch of philosophy that deals with the nature and value of art objects and experiences. It is concerned with identifying the clues within works that can be used to understand, judge, and defend judgments about those works. Originally, any activity connected with art, beauty and taste, becoming more broadly the study of art's function, nature, ontology, purpose, and so on.'*

The 'art experience' is interesting from an artistic point of view, but can not be formalised without making bold assumptions. Because this project is not purely about "Art for art's sake", we will focus especially on the functional and formal side of art. Meaning will generally be less important. The following is a refinement towards a more exact definition of aesthetics:

[1] Taken from http://www.artlex.com.

> 'Art is the imposing of a pattern on experience, and our aesthetic enjoyment is recognition of the pattern [39].'

Here, an objective relationship is made explicit between art and experience of it. This approach is taken a step further in [6], where the notion of an *aesthetics of particulars*, or *exact aesthetics* is discussed, especially in relation to modern art. Here it is stated:

> 'The distortions and reconstructions of the dynamic process of vision in aesthetics are not in fact subjective. They are *phenomenologically objective*, [...]'.

According to Albertazzi, there are two fundamental scientific approaches to understanding aesthetics: the first is that of philosophical theory (which we have discussed to some extent in this section). The second is *Gestalt theory*, a subdiscipline of perceptual psychology, on which we will focus below.

Gestalt Theory As is pointed out in [9], at the end of the nineteenth century, a tendency within the art movement towards *aestheticism* took place: the underlying structures and forms of different art forms were seen as the essential basic, most abstract representation. A lot of the ideas of early aestheticists can also be traced back to Ukiyo-e art. These were based on aesthetic principles that emphasised balance, minimalism and implicit beauty that required mental participation of the viewer to become explicit. When the twentieth century kicked off, several books had been written about Japanese composition, which had become very popular (for further reference see [9]). The ideas about composition started to become common knowledge. Aestheticism started to reflect on other artistic disciplines such as, for example, architecture (examples include Lloyd-Wright, 'Bauhaus' and 'De Stijl'). Aestheticism even started to have a less apparent influence on perceptual psychology.

Gestalt theory was developed by a group of German psychologists during the 1920s and 1930s[2]. They claimed that the separate analysis of each of the senses was not sufficient to explain high-level sensory experience. Instead they took a holistic approach, believing that the combined experience of different stimuli in the same picture was more powerful than the sum of the experiences of the separate stimuli. Thus, the concept of a 'good figure' was developed, which in short says that the human eye always tends to perceive one particular, optimal, form. This was called the *Law of Pragnanz*. The Law of Pragnanz is fundamental to Gestalt theory.

There are many principles underlying the Law of Pragnanz. Many of them are more relevant to psychology than to visual art and design. In 1923, the founding member of gestalt theory, Max Wertheimer, wrote an essay entitled "Theory of Form" [38] in which the basic ideas of gestalt theory related to visual language and perception would be laid out. Once picked up by educators in arts and design a couple of decades later, these ideas would prove a formal validation for the theory behind composition and graphic design.

The Gestalt laws that are related to visual perception are such common knowledge that it is easy to find many sources of information. In [12], for example, they are

[2]For futher reference see the International Society for Gestalt Theory and its Applications (http://www.enabling.org/ia/gestalt/index.html).

analysed in the context of visual screen design. Because this document is fairly recent and extensive, we will handle the relevant gestalt principles based on it.

Symmetry or *balance* is achieved by placing the visual elements in even distribution on either side of an axis in the visual field. Note that most of the gestalt principles are still general principles which can be interpreted in different ways. An image could be balanced over the vertical direction, but not in the horizontal direction, for example. Or an element which is relatively large can function as a 'counterweight' for a group of smaller elements on the other side of the axis.

Good continuation states that a visual field element or a group of elements can be shaped in such a way that the observer perceives directionality. This causes, almost forces, the eye to follow a path along it.

Closure or *completion* states that missing parts of a shape will automatically be added on perception. The picture is completed or 'closed' inside the mind.

An interesting distinction is made between the *figure* and the *ground* of a visual image. The figure is the part of the image that is seen as the 'object' in the front, whereas the ground, or background, is the rest of the surface of the canvas that is behind the figure. Because depiction is done in the flat plane, a borderline holds the figure and the ground in equilibrium. When beholding the canvas, the shape within this border and that outside it are complementary. Psychologically, the interesting aspect of the figure-ground principle is that figure and ground can be switched, turning the figure into 'mere' background and the background into a particular object. In figure 2.2(d) for instance, a picture of a vase is shown. But when the shape of the vase is abstracted, the complementary ground shapes appear to form two opposite faces.

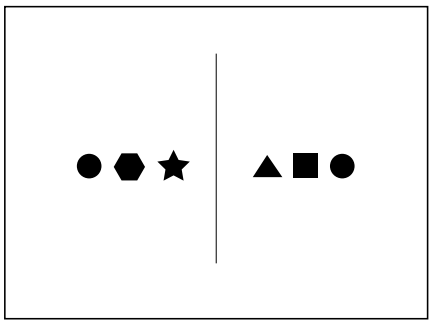
According to gestalt theory, in every visual field exactly one maximally salient point is needed, a *focus point*. The viewer's attention is attracted by it and once it is perceived the eye of the viewer is led by the visual flow of the image. The idea of a visual flow is not explicitly formulated as a gestalt law, though.

The principle of *proximity* states that grouped elements will be perceived as belonging together. In other words: grouping. Another principle that is considered important in the context of grouping is that of *similarity*, which states that similar objects will be mentally represented as part of the same visual cluster.

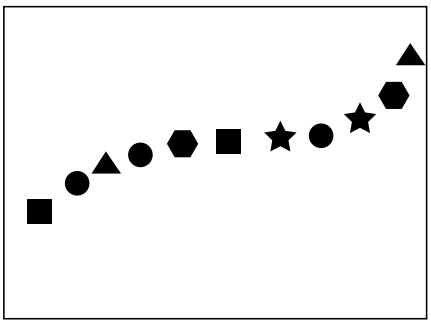
Finally, *harmony* or *unity* states that elements that are bound together acquire a new form. Elements that are not part of the group are considered unrelated, which can lead to distraction and confusion. This principle can be seen as a visual variation of harmonics in music theory. Another aspect that has considerable influence on harmony is the use of colour In [32], research has been done as to what kind of colour-schemes are aesthetically pleasing. This has resulted in the *Cuypers Colorpicker System*.

All the gestalt principles described here are concerned with a two-dimensional environment, but how do they hold for depth? We will not make use of a three-dimensional model, but we will consider depth effects in the flat plane later on. Depth effects in a composition, known as *pictorial depth perception* is achieved by overlapping shapes on the one hand and varying the size of objects. In the case of grouped objects, depth can be achieved by applying these tricks to whole groups instead of single shapes. Practical principles of pictorial depth perception are discussed in the next section (section 1.4).

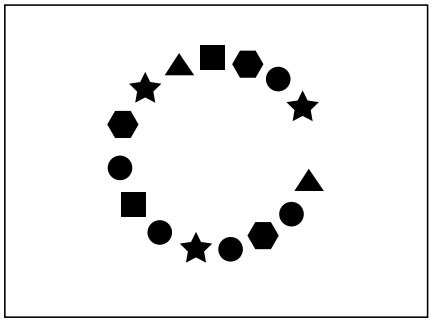
Gestalt theory can provide a basis for the analysis of visual compositions and as is discussed in the next paragraph, its principles have a sound connection with the ele-



(a) symmetry/balance

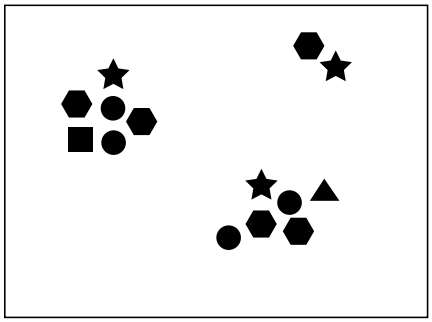


(b) continuation

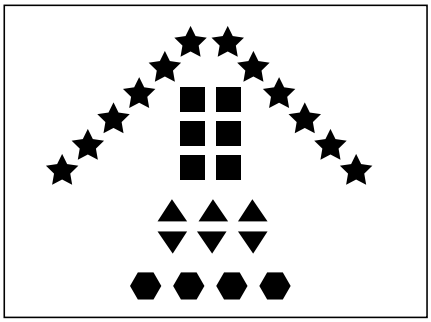


(c) closure

(d) figure vs. ground (Rubin's vase)



(e) proximity



(f) harmony

Figure 2.2: gestalt theory principles (adapted from [12])

mentary ideas behind graphic design. Theoretically there are several points on which it is criticised: first, experimental data and conclusions derived from them are less straightforward than those of psycho-physical approaches to visual perception. Second, there is uncertainty about the way abstract patterns should be interpreted in terms of form perception. On the other hand, after almost a century the main principles of gestalt still stand and recently much new research is being done in the area of *neo-gestaltism* [34].

Practical Graphics Theory By taking a top-down approach, we started by briefly discussing aesthetics, the philosophy of art, and after that we laid a theoretical basis for the analysis of pictorial perception. Finally, we will approach art from a practical angle by looking at an artwork from the point of view of the creator.

The Composition As was shown in the section dedicated to the history of modern art, section 1.1, there are many different kinds of art. The most basic, generic description of a two-dimensional artwork is called a *composition*. A composition of a painting can be seen as a formal framework that specifies what a 'well-formed' artwork at the highest abstraction-level looks like. There is only a limited number of possibilities for arranging objects in a flat, rectangular plane in a conceptually distinct way. Throughout history, several forms of composition have proven to be effective. Based on [36], we will give a short explanation of the most relevant ones.

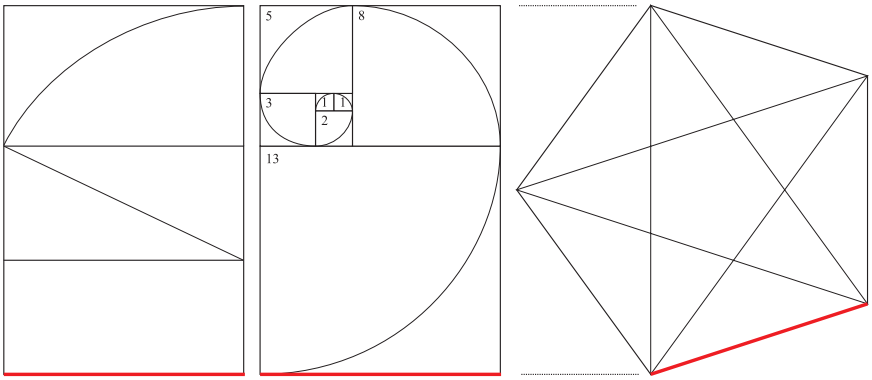
Let us start by considering the basic features of a composition. An artist who wants to make a painting will first of all need something to paint on: the *canvas*. Within the boundaries of the canvas, the elements of which the artwork consists are depicted. These elements can consists of anything: objects that are animate or inanimate, realistically drawn or abstract, detailed pen strokes or wild streaks of paint. Of course, this is all highly dependent on the style or particular art movement. Now the question is: how should the elements be arranged? The answer to this can be found in examples of fine art throughout history.

In still lives, for instance, a couple of inanimate objects (for instance pieces of fruit or pottery), are placed together in a composition. The still life, in fact, doesn't need meaning, it is an exercise in form. What distinguishes the still life by the master painter from one painted by the art student, apart from superior painting skill, is how the painting initially 'grabs' the attention of the viewer, how attention is gradually 'led' through the painting and how natural the distances and alignments between the composed objects are.

The most straightforward composition is the *symmetric* composition, in which identical objects are arranged in such a way that they are mirrored in a vertical axis. This way, the distances to the left and right of the axis are always equal which results in a natural, balanced whole. In an asymmetric composition, on the other hand, the choice of horizontal and vertical measurements between objects is not so straightforward. Here, we have to make use of certain *proportions* to keep the composition in balance. Since the time of the Greeks, artists have experimented with ways to let constructions such as temples look well-balanced and be structurally sound at the same time. This has led to the development of several geometric constructions that specify certain proportional measurements, among others the *diagon*, the *quadriagon*, the *hemidiagon*. However, the *auron*(fig. 2.3(a)), the proportions of which are better known as

the *golden ratio*, is the most significant.

The golden ratio is often denoted as Φ (capital phi) and its value is approximately 1.6180339887. Φ is an irrational number that occurs in nature similarly to the constant π. It can be found in the proportions of, for example, plant leaf patterns, the spiral shape of shells and the exponential growth of a population. Important renaissance painters such as Leonardo DaVinci rediscovered the golden ratio and applied its measurements to their paintings, believing to have found the ultimate proportional balance (the golden ratio is also called the *divine proportion*). Mathematically, the golden ratio is interesting because of the many ways it can be found. The Fibonacci numbers (0,1,1,2,3,5,8,13,21...), for example, represent the growth pattern of a population. The ratios of the pairs of successive numbers converge to Φ. By aligning squares with measures that correspond to the Fibonnaci numbers in the way depicted in figure 2.3(b), drawing a curve through the corners will create a perfect spiral. The shape of this spiral also regularly reoccurs in nature. There are many other mathematical methods to derive the golden ratio, for example, by dividing the length of a line between two non-neighbour points with the length of one of the sides of a regular pentagon (figure 2.3(c)). Any rectangular area can be infinitely divided into parts that

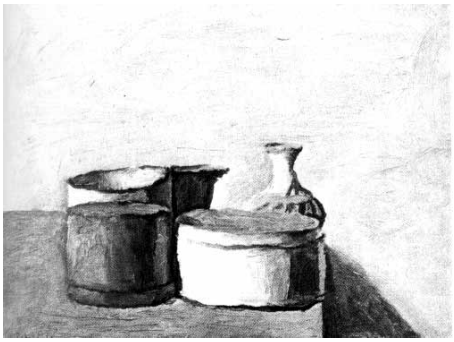


(a) auron (b) fibonnaci numbers (c) pentagon

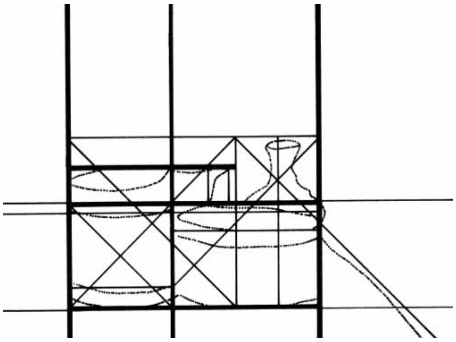
Figure 2.3: three ways of finding the golden ratio

are proportioned according to the golden ratio. In figure 2.4, for example, a schematic decomposition (figure 2.4(b)) is depicted next to the original artwork (2.4(a)), which is based on the golden rule proportions.

Important asymmetric composition types are the *diagonal*, *triangular*, and *geometric* composition. The diagonal composition is based on a single diagonal line around which the important elements are aligned. An example of a diagonal composition is depicted in figure 2.5. In figure 2.5(a), the painting is depicted in it's original form. The main figures are composed around a diagonal line from the top-left to the bottom-right corner. The mirrored version of the painting is depicted in figure 2.5(b). This results in a change of the dynamics of the composition. In the first, the composition is leading downward, in contrast to the second, which seems to be more uplifting. This



(a) still life, Giorgi Morandi



(b) decomposition of golden rule proportions (adapted from [36])

Figure 2.4: use of golden rule proportions in fine art

difference could maybe be contributed to the fact that in Western cultures people have learnt to read from left to right. It could be that the dynamics of the composition could be perceived differently in cultures where the reading direction is different.

The *leading line* of a composition is an imaginary line which follows a path through the most salient elements that are close to the focus point. This has the effect of continuation, which, as was mentioned earlier, is one of the important principles of gestalt to visual design. Therefore, in a diagonal composition, the leading line can be said to coincide with the diagonal line. Compare this to the symmetric composition, in which there is no explicit line along which objects are placed. The *triangular* or *pyramidal* composition has a leading line which encloses a triangular area within the borders of the canvas. Some examples are depicted in figure 2.6.

The name of a geometric composition is somewhat ambiguous in the context of the topic of this thesis. It is actually concerned with the most extremely abstracted compositions, for example Mondrian's later works, in which no actual objects, or even shapes, are depicted[3]. Other important composition types are the ones which a ragged or curved leading line (sometimes denoted as *movement* compositions because they give a dynamic impression) and *over-all* compositions, which do not have a singular point of attention but instead consist of patterns of objects or shapes that cover the entire canvas.

Depth composition is an important aspect of painting. Figurative paintings must often be perceived as being three-dimensional, although they consist of a flat surface. Arranging the pictorial elements in the right way can let some of them seem to be placed at the back, while others seem to stand right in front of you. This is called *linear perspective*. The three basic principles behind linear perspective are the *layering* of objects, the *disappearance point* and the *gradual decrease of size* the further away an object is placed. Overlapping objects will seem to be stacked, where the top object is closest and the object at the bottom will be furthest away. The size and the

[3]Geometric compositions sometimes make use of *modular* proportions instead of the golden ratio, which means that every measurement is a multiple of some base value.

(a) normal (b) mirrored

Figure 2.5: diagonal composition: 'The Elevation of the Cross', Rubens

(a) 'The Martyrdom of St. Sebastian', Pollaiulo (b) self-portrait, Rembrandt

Figure 2.6: examples of triangular compositions

distance between the center of the objects determine how far away from each other they seem. This is called the *interposition of the objects*. The further away, the closer to the *disappearance point* the object will be. When an object is close, it will tend to move outwards in the horizontal and vertical direction the further away from the center of the view it is. Depth is achieved in the flat plane of the painting through careful arrangement of the composition. This results in a diagonal or triangular composition in the flat plane. Interposition is a straightforward way to describe the level of depth complexity and arrangement properties. Varying the interposition of objects can cause different kinds of depth-effects [36].

Graphic Design *Graphic design* is another relatively formal discipline of art theory. Graphic design and composition are in fact closely related to each other, but whereas composition is focused on a descriptive set of visual arrangements, graphic design is based on a couple of principles which can help to guarantee the quality of a composition. Graphic design is traditionally applied to the formatting of page-layouts, but is also fundamental to graphic art and especially to two-dimensional vector-based art, which often has a graphic design look-and-feel. Graphic design is also closely related to perceptual psychology. When gestalt theory was developed, it was accepted by the design community as a scientific validation of age-old ideas. Whereas gestalt is analytical by nature, graphic design is focused more on practical guidelines to create good layout, typography and art in general. We will discuss the principles of graphic design based on [40].

The first basic concept of graphic design we will discuss is called *emphasis*. Every piece of art has something to say, a point, be it literal or abstract, or something in between. This point is almost always conveyed by means of a central place in the painting, a visual element or group of elements around which all else is composed (an exception to this is of course the over-all composition, in which all elements can be said to be equal to each other). This point is called the *focus point*, which has an obvious parallel to the Gestalt focus point mentioned earlier. Furthermore, emphasis is achieved by *contrast* and *isolation*; the visual elements around and nearby the focus point are placed relatively further away from other elements or they contrast in color, shape or size. This separates the group from the rest of the artwork.

The theory of composition explained above corresponds to the concept of *balance* in graphic design. Not surprisingly, balance is similar to symmetry or balance in gestalt theory, but it also corresponds to ideas about composition; a good composition is a balanced composition, so to speak. But we can also look for balance at a more detailed level: the figure-ground principle, for example, can be applied to an entire canvas but also to a smaller bounded area.

Rhythm is based on repeating elements. Just as with the leading line principle, the elements of a visual rhythm pattern can guide the eyes of the viewer through the painting. In fact, often several rhythmical structures of visual elements are aligned in the direction of the leading line. But, dependent on how full the composition is, visual rhythms can also be found throughout the rest of the painting in a less prominent role. Flowing, connected rhythms will give an impression of harmony, while at the other extreme a divided distribution of similar objects will give a dynamic impression[4].

[4]This can seen as a visual analogy with rhythms in music.

The idea of *unity* is centered around the observation that "the whole is bigger than the sum of its parts". Unity binds all the loose concepts of graphic design together. There are three principles underlying unity: *proximity*, which is the same as grouping. *Repetition* of properties, visual elements or techniques. Rhythm can be seen as a form of repetition. Other examples are repetition of colour, repetition of texture and repetition of shape (it should be noted that in our approach visual elements and shapes are basically the same). *Continuation* states that unity can be achieved by aligning visual elements along lines, edges of a shape or the direction of another element, resembling the continuation principle of gestalt theory. This principle is often applied to the lay-out of magazines and websites.

2. Artificial Intelligence Perspective

We have analysed some ideas in the previous section which can be useful to build a formal framework for the representation of an abstract geometric artwork. Because we want to generate different artworks (semi-)dynamically, our second goal is to be able to adapt and switch between representations. This means that we need a component that can control this process; the back-end of the system which explicitly or implicitly contains our problem solving strategies.

In this section we will discuss several disciplines within the field of AI which could provide a basis for this component. These include quantitative and qualitative spatial reasoning, constraint-based reasoning and machine learning.

2.1 Reasoning

The most straightforward AI approach to building a system would be by implementing a set of rules which form a reasoning component. Rule-based reasoning systems are based on logic [30]. Non-formal ideas are formalised by translating them into descriptive logical sentences. In our case, we would need a reasoning component which is able to describe two-dimensional vector graphics. More specifically, we would need a form of two-dimensional spatial reasoning in order to write rules that describe our knowledge representation.

Spatial Reasoning A logic-based system is built up out of propositional symbols, the truth symbols *true* and $false$ and the connectives $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $=$, which respectively stand for *conjunction*, *disjunction*, *negation*, *implication* and *equivalence* [10, 30]. Together they can form sentences of propositional calculus. When we want to denote a certain spatial property, we can define a proposition such as "the circle is under the square" or "the square is left of the triangle", which can be true or false. The propositions are denoted as atomic symbols, for example P and Q. Combining the two by using one of the connectives, for example a conjunction, yields a composed propositional sentence: $P \wedge Q$ which would mean "the circle is under the square and the square is left of the triangle". Now, a new relationship has emerged; because when the circle is located under the square and the square is left of the triangle, the circle also must be left of the triangle (assuming that the circle is directly under the square). This would introduce proposition R, which cannot simply be derived from the previous propositions.

Predicate calculus provides a more flexible solution. Sentences in predicate calculus can be composed in the same way as propositional sentences, using the same connec-

tives [10]. However, relationships between atomic entities can be defined explicitly. Instead of defining each relationship over and over again in a separate proposition, a general relationship or *predicate has_position*(*Object*, *Position*) can be defined, in which the *term Object* can be instantiated with any object in the world and the term *Position* can also have any predefined value. Predicate calculus is the most commonly applied paradigm in logic programming (although there are various more exotic approaches, for example temporal and epistemic logic [31]). Therefore, if we choose to implement our system in a logic programming language, such as Prolog, we will have to distinguish the atomic entities and relationships involved.

Representation of atoms and relationships is not always straightforward. Colour, for example, can be represented in several ways [32]. There is the traditional colour-coding scheme that is used for printing, CMYK (Cyan Magenta Yellow blacK), which is conceptually not the most suitable for screen-colour representation since colour monitors generate colours by (additively) emitting light instead of (subtractively) applying ink to a white surface . There are two candidate additive colour representations: RGB (Red Green Blue) and HSL (Hue Saturation Lightness). Of these two, RGB is the most common, but is purely descriptive, while the HSL model is a much more conceptual representation of colour and has been successfully applied in [32] to implement a system that generates colour sets based on user interface requirements. We will handle colour in further detail in 2.3.

For our project the most important atomic entity is probably the one that specifies shape. In a sense, shapes form a 'visual vocabulary' to convey the structure of the composition. The traditional mathematical representation of geometric shapes consists of *points* and *lines* (called *nodes* and *edges* in graph theory terms [35]). But as is argued in [17], a shift to a higher-level description of geometric concepts can give the domain description a more conceptual twist. A second candidate representation of atomic objects would in this case be the 'shape'-entity: a geometrically styled painting must inherently be built up out of one or more geometric objects. Shapes can vary from basic rectangles, circles and triangles to complex non-convex polygons or shapes that have curved edges.

We are primarily interested in the artistic composition of geometric forms. As was explained in section 1.4, the way that shape objects are grouped and aligned is important in this respect. We will therefore focus on *spatial relationships* between shape objects. However, the geometric properties of individual shape are important as well. When a composition consists of squares, replacing these squares with circles would not change the locations and alignments of the objects, but the interaction between the objects and the whitespace between them, the figure-ground principle, would be entirely different.

In a domain such as architecture there is often an apparent mutual dependency between shapes and transformations. Take for example a staircase. In one pattern, rectangular objects are arranged in a spiral. Replacing the objects with circles would change the fundamental aspects of this pattern, disrupting its function. Thus, the properties of a shape and the role of the shape in a larger context are by default dependent on each other, although in a more 'vague' domain such as ours this is less important. Because our project is not based on geometric art compositions, dependencies are mainly based on geometric characteristics.

2.2 Shape Patterns

If we look at a collection of atomic shapes as a 'visual vocabulary', by analogy, conveying the underlying ideas of our abstract composition can be seen as communicating in a certain 'image language'. A field of research that is based on this point of view and is relevant to reasoning with shapes is that of *shape grammars* or *shape algebras* [13, 24]. Research has been done to analyse the relationships between different forms of shapes, switching between the shape representation and the traditional point-line representation in a morphology of two-dimensional graphics. This is classified as the *subshape* problem.

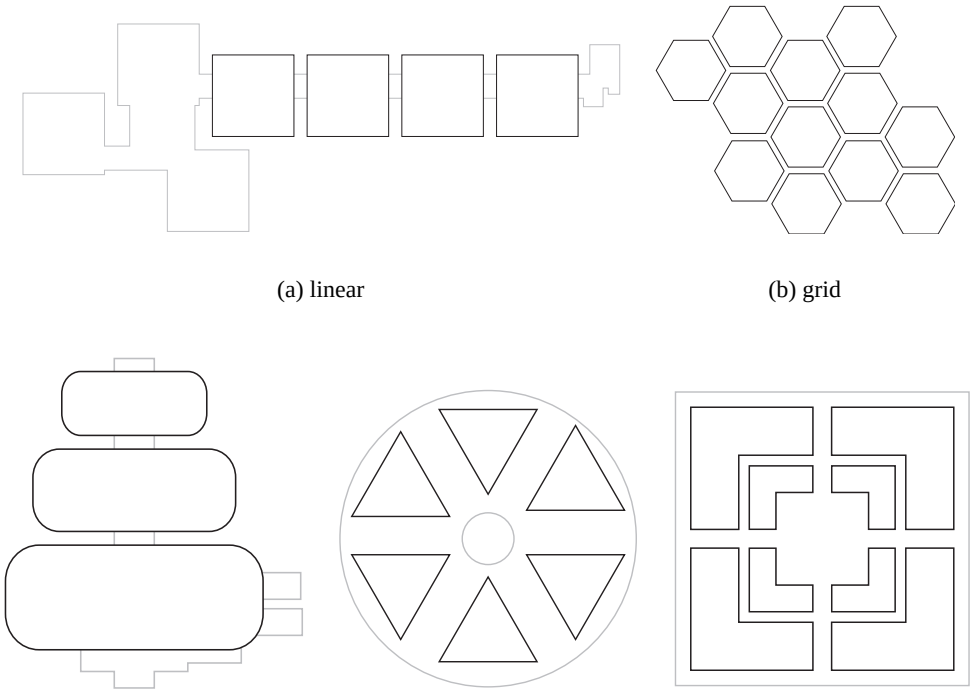
Interesting research has been done in the same field in higher-level constructions of shape grammar elements, called *shape patterns*. In [11], forms are classified as *(composite) shapes*, *subshapes*, *primitive shapes* and *group shapes*. Primitive shapes are atomic, they cannot be decomposed into smaller objects. Composite shapes are built up from primitive shapes, which are then called the sub-shapes. Shape patterns include shapes as well as spatial relationships between them. The ideas behind shape patterns can be related to the design pattern community, which places a relationship between software architecture and the decomposition of architecture into a finite set of patterns by Alexander [1, 2, 3]. The designs of artifacts or buildings such as Frank Lloyd Wright's *Prairiehouse* are taken as examples to derive structural algorithms, that can then be applied to model new constructions. Quantitative as well as qualitative topological relationships can be used to describe group transformations. Shape patterns and shape grammars are used in a wide variety of application areas, for example industrial design, archaeology, architecture and planology[5]. Our prime interest is in the combination of shape patterns with geometry.

According to [11], *'a pattern is a design in which a certain shape is repeated many times'*. A distinction is made between similarity in shape and similarity in spatial relationships. An instantiated hierarchy of different shapes that are arranged according to one or more types of spatial relationships is called a *schema*. The shape pattern is the abstracted knowledge contained in such a schema. The point of this is that different shapes can be arranged according to the same kind of spatial relationships, but also different kinds of spatial relationships can be applied to similar shapes. Because patterns are defined recursively they can be nested, which can result in more complex structures.

In more traditional architecture literature patterns are described in terms of *formative ideas*. Based on historical buildings, many distinct types of patterns can be recognised. Although architecture is fundamentally about the design of three-dimensional structures, many formative ideas can be based on blueprint representations; they have been projected on one of the perpendicular two-dimensional planes. Several formative ideas that are based on fundamental architectural ideas have been depicted in figure 2.7. Each of them is based on one shape. However, some additional shapes have been added, which do not belong to the particular pattern (greyed out) to indicate that the pattern is an abstraction from the entire scheme.

Figure 2.7(a) depicts a *linear* formative idea, which simply consists of a repeated unidirectional translation. In figure 2.7(b) a second dimension is added to the pattern. In addition to being translated, in figure 2.7(c) the shapes are also scaled. This is called

[5] For more information, see http://www.shapegrammar.org.



(a) linear (b) grid

(c) additive (d) circular (e) nested

Figure 2.7: some formative ideas (adapted from [11])

an *additive* formative idea. In figure 2.7(d), shapes are rotated instead of scaled. In figure 2.7(e), a composite structure of four shapes, each of them rotated 90 degrees, is copied and subsequently scaled, which results in a *nested* structure. From these examples it can be derived that both transformations as well as shapes can be composed into groups. This way, simple patterns are combined into more complex ones. Thus, just like a schema, a pattern consists of a hierarchical set of shapes as well as of a hierarchical set of spatial relationships between the shapes. In chapter 3, we will map ideas about shape patterns to our own conceptual model.

Quantitative Spatial Reasoning We make a distinction between two different approaches to reasoning: *quantitatively* and *qualitatively*. Whereas quantitative reasoning is based on precise, discrete values, qualitative reasoning is focused more on the relative relationships within the domain. We will first discuss quantitative reasoning about spatial characteristics.

The quantitative approach to spatial properties is based on a mathematical representation of space. This type of reasoning is often applied in the field of robotics, which is an obvious choice because computers are particularly suited to deal with large sets of calculations. The classic mathematical way to divide space into measurable real number coordinates is by using the *Carthesian coordinate system* with which any point in space can be represented on the basis of a defined reference point and two or three perpendicular axes. Carthesian coordinates are applicable to two-dimensional as well as three-dimensional space. They can provide a convenient notation form for graphics in general and in our case specifically for visual art, comparable to the notation of notes in music [22].

Some of the most important mathematical fields with respect to spatial arrangements are trigonometry, geometry and topology. They are often applied to *spatial analysis*, a practical discipline that is concerned with the analysis of spatial data such as topographic maps or camera data. Spatial analysis is sometimes done using reasoning, especially by Geographic Information Systems (GIS). Properties such as *convexity*, *concavity* and *overlapping* of surfaces have been thoroughly investigated. These properties can be used to formalise shape properties as well as spatial relationships.

Although three-dimensional graphics are beyond the scope of this thesis, it is worth mentioning that it is a highly developed research field within computer graphics. Some notions of two-dimensional space can be translated directly into parallel ideas in the third dimension by adding a third variable for the depth plane. Often however, entirely new concepts are introduced, for example lighting and normals. An important application area of three-dimensional spatial reasoning is Computer Aided Design (CAD) of architecture and interior design. For more information we refer to [5, 15, 21].

An exact spatial notation enables the formalisation of precise inferencing algorithms. Carthesian space is continuous, infinite and is highly suitable for mathematical formulae and scientific applications. But in some cases, the advantage can turn out to be a disadvantage. The level of precision that a quantitative approach provides is sometimes not necessary or even too high and a trade-off with a more conceptual notation is possible.

Qualitative Spatial Reasoning Qualitative reasoning is an approach that is much more based on ‘common-sense’. Objects are not measured in exact units but instead

are described in relation to each other. This is, from an artificial intelligence point of view, a big advantage, because it resembles the more ambiguous, but at the same time more flexible, human reasoning. James Allen's *interval algebra* [4] is a classic form of qualitative reasoning in temporal logic. Based on the time intervals, a classification of relationships is made [23, 29]. Intervals are denoted as pairs of real numbers, $[x^-, x^+]$, with $x^- < x^+$. The relationships include *before*, *after*, *during* and *equals*. The interval algebra can be applied in a spatial context, but because time has only one dimension and a spatial environment has at least two, matching the two can cause problems. This problem can be avoided by doing independent calculations for width and height (and in a three-dimensional setting depth).

Common two-dimensional qualitative spatial relationships are *above*, *to the left*, *inside*. When properties are described in terms of relationships instead of measurements, they will explicitly model the concepts they represent. An example of a qualitative description of spatial properties is the *Region Connection Calculus (RCC)* [17, 18]. This is a first order predicate logic that is based on the *connectedness* of shapes, which means that they share a common point. The RCC-8 calculus specifies eight different qualitative relationships: *disconnected*, *externally connected*, *partially overlapping*, *equality*, *tangential proper part*, *non-tangential proper part*. Of these last two there also exists a non-symmetric inverse relationship. These binary relationships for pairs of regions are termed Jointly Exhaustive and Pairwise Disjoint (JEPD). The RCC-8 calculus can easily be extended to describe more complex situations. The *egg-yolk* calculus for example, is an extension of RCC-8 to 252 JEPD relationships [18, 19]. On the other hand, the RCC-5 calculus consists of a subset of the RCC-8 relationships [26].

The ambiguity of qualitative reasoning can also be an important disadvantage; this is easy to see with the position-example of section 2.1. We have assumed that the 'under'-relationship was strictly one-dimensional, so the circle would be located directly under the square. There is no coordinate or measurement that specifies how far from the square the circle actually is, so the 'under'-relationship is in this case a qualitative relationship. But if 'under' is interpreted as being independent of horizontal directionality, it could also mean 'under to the left' or 'under to the right'. When reasoning is done without regarding the shortcomings of a qualitative representation, faulty deductions could be made.

3. Web Perspective

Domain knowledge and AI techniques can provide a background for a theoretical model of a generative art system. Apart from creating a formal model, we also want to build a prototype based on it and so we will need a front-end component. By front-end we meant the part of the system that contains the translation steps towards the final output format, as opposed to the back-end knowledge component that makes the decisions on composition.

There are several different technical options for implementing the front-end. There are widely available graphics libraries for many common programming languages such as Java and C++. A possible candidate is the Java 2D API. The functional language Haskell also supports two-dimensional graphics. Its graphics library Haven is based on Java 2D, although it offers a more advanced model of scalable vector graphics. Also, proprietary software, such as Adobe's *Live Motion*, Macromedia's *Flash* and

Director and Cycling '74's *MAX/MSP*, is available with which graphics can be edited and generated through scripts. However, although these would all be valid choices, neither of them provides the support for reasoning that a logic programming language does. Since our main perspective is that of AI, we chose to implement the back-end in Prolog. For the front-end part only an implementation of a suitable 2D graphics model is needed. We should be able to express concepts such as shapes, colours and transformations in terms of this model. Also, tools should be available to view and edit the output format. This has led us to the work of the World Wide Web consortium (W3C), which develops recommendations for various fields related to the internet. Of these, the Scalable Vector Graphics (SVG) format seems to sufficiently describe the common concepts of 2D graphics for our purpose.

3.1 Markup Languages & Standards

The internet has traditionally provided poor support for graphics and formal modelling. The HyperText Markup Language (HTML) with which websites were encoded has been a driving force behind popularizing the web because of its simplicity, but at the same time has limited more professional applications of web technology. This has been the main reason to develop the eXtensible Markup Language (XML), which is a subset of the more structured Standard Generalised Markup Language (SGML) (on which HTML was originally based) [33]. With XML, domain-specific data can be inherited using so-called Data Type Definitions (DTDs). Thus, XML is independent of content and XML definitions can specify many different kinds of functionality. These include:

- semantic annotation (RDF, RDF Schema, DAML+OIL, OWL)
- multimedia (SMIL)
- vector graphics (SVG)

3.2 Vector Graphics

A computer screen consists of a raster of picture elements (pixels), each of which is assigned a certain colour. Anything that is displayed on it, be it a text, a part of the user interface or a full-colour image, has to be encoded in pixels. Basically there are two ways to do this: by storing the (relative) position of each colour in a *bitmap*, or, more generally, encoding a model of the things to be displayed as well as a way to translate this model into a rasterised image. Although this last approach demands a more complex representation and more realtime computations, it is also more flexible (scalable, easy to combine) and in general takes up less harddisk space. Another advantage of realtime rasterisation is that it is easy to create animated graphics. Whereas a bitmap animation consists of a linear sequence of consecutive bitmap images, animation of vector-based graphics (3D as well as 2D) can be created simply by changing some parameters of the model each time just before the image is rendered. Standard image formats such as JPEG, GIF and PNG fall into the bitmap category, while fonts, two-dimensional (2D) vector graphics and three-dimensional (3D) graphic models are rasterised. Fonts are generally a subset of 2D vector graphics.

Several popular formats for non-animated 2D vector graphics exists. These include Encapsulated PostScript (EPS) used for embedding graphics in printable documents,

the Portable Document Format (PDF), which is similar to EPS but available as freeware, and the Adobe Illustrator (AI) format. Macromedia's animated 2D vector format ShockWave Flash (SWF) is distributable and several applications, such as Macromedia Director, Macromedia Flash and Adobe Live Motion, also can export to SWF files. A browser plugin is needed to be able to play SWF files. Adobe Illustrator 10 supports scripting in several languages, including JavaScript and Python. *Actionscript*, which is a scripting language for SWF files that is based on ECMAscript, is widely used.

While EPS and PDF were primarily designed for document lay-out, the more flexible SWF is a closed source format. Adobe Illustrator is a proprietary software package. The open web standard SVG provides a good alternative. As mentioned before, SVG is an XML-based syntax for 2D (animated) graphics. Its functionality includes everything from basic shapes and transformations to Bezier curves and animation. We will further explain vector graphics principles based on the SVG specification [37].

3.3 Scalable Vector Graphics (SVG)

In fact, most 2D vector formats can also embed bitmap formats. SVG supports bitmap images as well typography. These can undergo most of the functions that SVG offers, but we are mainly concerned with general vector graphics: shapes and transformations.

First, we will give a short explanation of the concepts and the environment of SVG. As described in section 1.4, page 15, the metaphor that is commonly used for the active area in which graphics are rendered is that of a canvas. This is part of the *painters model*. Objects are layered on top of each other. The solid parts of the upper layer hide the parts under them. Instead of solids, one can also make use of *alpha channels*. This is an extra channel that is added to the colour model (section 2.1) which specifies transparency. Colour can be specified by an exhaustive list of keywords (for example 'red', 'lightsteelblue', 'mediumspringgreen') or by a hexadecimal code which specifies a certain RGB value (this a common way of encoding colour values on the web). Colour is part of the *Paint* class. Paint can be applied to the inside of a shape called the *fill* or to the *stroke* of its lines. Apart from solid colour, paint can also consist of a linear or circular gradient, a pre-defined (bitmap) pattern or it can remain empty, in which case nothing is rendered.

A distinction is made between *basic shapes* and *paths*. The set of basic shapes contains the most typical geometric forms, *rectangle*, *circle*, *ellipse* and *line*. The general *polygon* and *polyline* (a polygon path that is not closed) classes are also considered basic shapes. The polygon class can specify triangles, rectangles, pentagons, hexagons, et cetera. Each basic shape has a standard set of attributes. We will give a short summary.

The rectangle basically requires 4 attributes that specify shape, `x`, `y`, `width` and `height`. Because the origin of the coordinate system is in the top-left corner of the canvas, the positive y-axis is directed downwards. Thus, `x` and `y` are the coordinates of the top-left corner of an object:

```
<rect x="40" y="20" width="20" height="10" fill="red"
stroke="blue" stroke-width="2"/>
```

The circle and ellipse objects are basically the same, except for the fact that ellipse has two fields for the radius instead of one. Both shapes have an `cx` and `cy` coordinate for the center point:

```
<ellipse cx="40" cy="20" rx="20" ry="10" fill="red" stroke="blue"
stroke-width="2"/>
```

The coordinates of a polygon or polyline are specified as a list of consecutive points. Points are denoted as comma-separated pairs of numbers, which in turn are separated by spaces. There is no difference between the two, except that when the first and the last coordinate in the point list are different, the path of the polygon will automatically be closed. The line element is conceptually the same as the polyline and polygon, but only specifies two points. Therefore, a line has four separate variables for its coordinates instead of a point list. Obviously, a line is never filled.

```
<polygon points="40,20 80,60 0,60" fill="red" stroke="blue"
stroke-width="2"/>
```

```
<line x1="40" y1="20" x2="80" y2="10" stroke="blue"
stroke-width="2"/>
```

Any non-curved geometric object can also be described in terms of a polygon. *Paths*, on the other hand, provide a similar representation of shape objects, but can also specify different types of curves. These include *cubic Bézier* curves, *quadratic Bézier* curves and *elliptical arc* curves. Apart from the notation of fills and strokes, paths have a different syntax. A list of variables is specified as part of the `d` attribute (which stands for *path data*). This list is subdivided into several parts which are preceded by a letter which specifies the functionality. For example, an `M` is used to denote a *moveTo* to a certain coordinate, an `L` denotes a *lineTo* and a `C` a *curveTo* (these are absolute coordinates, the lowercase equivalents stand for the relative variants). A path element would look like this:

```
 <path d="M 10 10 L 300 200 L 400 100 z" fill="red"
stroke="blue" stroke-width="2"/>
```

the path is closed. `C`, indicating a cubic Bézier curveTo, can be used instead of `L`, but requires two extra coordinates for the control points. A polybézier can be formed simply by putting multiple sets of curve-variables in sequence. The principles underlying the other curve functions are similar to the standard *curveTo* function. There also exist simplified versions of *lineTo* and *curveTo* commands which take fewer parameters.

SVG offers 4 types of transformations, *translation*, *scaling*, *rotation* and *skewing* along the *x* and *y*-axis, each of which can be specified by adding a `transform` attribute to a group-element, <g>, which in turn encapsulates the transformed shape(s). For example, translation of a line looks like this:

```
<g transform="translate(10,20)">
    <line x1="40" y1="20" x2="80" y2="10" stroke="blue"
    stroke-width="2"/>
<\g>
```

Other values of the transform attribute can be `scale(factor)`, `rotate(angle)`, `skewX(angle)`, and `skewY(angle)`. Multiple transformations can be done by forming groups with transformations around shape objects, or by listing them in the transform attribute field:

```
<g transform="translate(10,20) scale(2) translate(5,12)">
   <line x1="40" y1="20" x2="80" y2="10" stroke="blue"
   stroke-width="2"/>
<\g>
```

Technically, representations and transformations for rasterised images are mainly based on linear algebra. Transformations are calculated with standard linear algebra algorithms. A transformation is represented as the following 3×3 matrix:

$$\begin{pmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{pmatrix}$$

Only the top six values are relevant for the specific transformation. The transformation matrix is also denoted as $[a\ b\ c\ d\ e\ f]$. The values of the new x and y coordinates are calculated according to this formula:

The values of the different types of transformations can be found in the appendix.

SVG offers several other interesting types of functionality which will not be applied here but are worth mentioning, most importantly interactivity and animation. SVG implements an extension of the Document Object Model (DOM) which is the standard framework for document structure in HTML and XML. This includes support of standard user interface events such as *mouseover*, viewing events such as *resize* and error handling. SVG specific functions include *zooming* and SMIL timing events such as *repeat*. Bindings are specified that link the Java and ECMAScript languages to the (SVG)DOM. Also, the SMIL element `animate` is specified for animation.

4. Summary

In this chapter we have analysed the background of several fields that are of interest to our research. In section 1 we discussed modern art. We compared the fundamental ideas of the expressionist art movement to scientific developments in psychology, which we in turn related to the practical principles of graphic design. In section 2 we discussed what common methods exist to reason about spatial properties and in section 3 we explained the basic ideas behind the W3C recommendation SVG, which provides a suitable output format to encode geometric graphics in. Together, these three perspective form a theoretical basis for the design of a conceptual framework for the generation of abstract geometric compositions. This is done in the next chapter.

Chapter 3
Conceptual Framework

Based on the domain theory, we have designed a conceptual framework. This framework provides the formal basis for our graphics generating system. It should be emphasised that at this stage we only focus on descriptive aspects of composition generation. We will show to what level ideas about the artistic content can be conveyed through this framework in section 4.

This section is split up into four sections: First, we will start by deciding how to incorporate the main ideas into an algorithm which can generate a compositional template in section 1. After this, we will map the domain knowledge to this template (section 2). We will then make a top-down decomposition of the framework for abstract geometric art we have created and analyse at which levels what kind of reasoning could take place (section 3).

1. Compositional Template

We will model our conceptual framework in terms of a structural skeleton. As a starting point, we will analyse an example composition. Our goal is to extend the ideas behind this composition with the rest of our domain knowledge and eventually being able to produce a similar result on the basis of AI algorithms. We designed a vector-based abstract geometric composition in a graphics editor, inspired by the ideas of the domain theory of section 1.4. The result can be seen in figure 3.1. First of all, we have chosen for an angular composition. There is a certain point, above and to the left of the middle of the canvas, to which the rest of the composition is oriented. Two points on either side of the canvas connect with this focus point to form the leading line. Furthermore, several groups of objects are layered on top of each other. Each group has its own distinct colour. We have chosen to use greyscale tones, because we wanted to keep the colour scheme as simple as possible for the time being. The shading of an object is less salient the more it is placed to the back (it is closer to the background colour). Within each group, the separate shapes are arranged in such a way that they slightly differ from each other and sometimes partly overlap. Also, the arrangement of the group can be thought to have a direction that corresponds to that of the leading

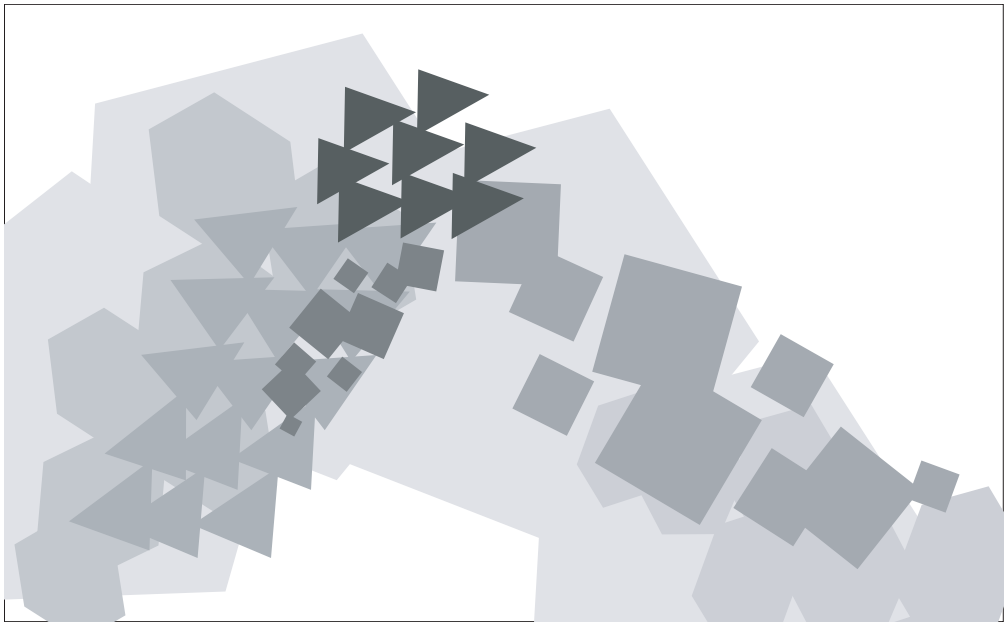


Figure 3.1: example composition

line of the composition. A sense of depth is achieved by layering the groups on top of each other. Because each group consists of a pattern of shapes, there are gaps through which the underlying layers are visible. We have chosen the simplest possible shapes, convex polygons, as a starting point. We have used polygons between three and six points.

To summarise, several ideas have been applied in this example:

- compositional style,
- the focus point,
- the leading line,
- a visual vocabulary,
- a basic colour scheme,
- layering of objects and
- shape patterns.

Together, they form a minimal set of ingredients for the generation of a composition. We can now look at properties of the graphics we want to generate at a more holistic level. Two important observations can be made:

1 The composition can be represented as the three elements canvas, focus point and leading line,

2 the structure of the composition can be decomposed to several levels of groups of objects.

When the exact values of canvas width, canvas height, the `x` and `y` coordinates of the focus point and the points where the left and right part of the leading line reach the canvas edge are known, a three-point line can be specified. The combination of these elements, as depicted in figure 3.2(a) can be seen as the most basic representation of the triangular composition, which explains our first observation.

To be able to represent the composition at this abstraction-level graphically, we will introduce *guides*. A guide is a concept that is often used in graphics tools[1] to make alignments more explicit. Conceptually they have an infinite length and an infinitely small width, which in practice means that they end at the edge of the viewing area and are rendered at the same (small) width at every zoom-level. Thus, when we model a composition graphically, we will represent it as a frame in which guides determine the alignments of the contents, with some additional information in the form of points at which guides cross.

1.1 Knowledge Decomposition

A decomposition of abstraction-levels for the knowledge domain can provide a solid formal basis for the conceptual framework we are designing. In our decomposition[2] we will distinguish three levels. The first level we distinguish we will call the *compositional level*. This is the highest level of abstraction and corresponds to the complexity-level of our desired end-result. The second level is that of *graphic entities*, several of which are needed as visual content for the composition. Each graphic entity can consist of one or more distinct objects. Because we have restricted ourselves to vector graphics, graphic entities can either consist of shapes or coherent shape groups. The lowest level, that of the coordinates of the points and lines of the shape, is less conceptually significant, but important from a technical standpoint. Because the formal description of shapes at this level corresponds to that of mathematical graphs, we will call this the *graph level*. The dimensions that are important for each of these levels will be discussed in section 3.

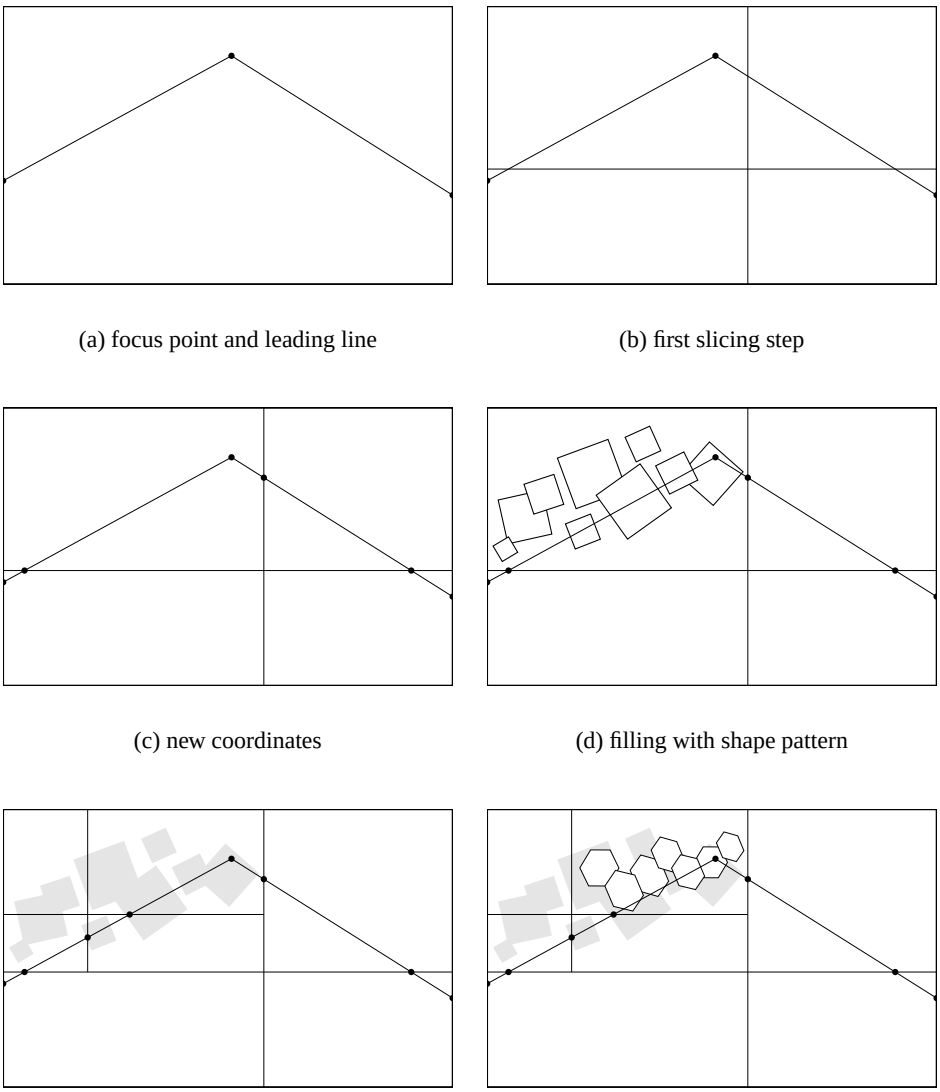
1.2 Recursive Model

Building a composition from scratch is a bottom-up task. By taking the general structure of the canvas, focus point and leading line, we can incrementally add graphical entities according to certain spatial relationships such as proportions and alignments. This can be defined recursively. In figure 3.2 we have taken the ideas mentioned in the previous paragraphs and designed a stepwise plan for constructing a compositional template. This is depicted in figure 3.2(a).

As shown, we started out with the basic representation of a composition, which consists of canvas, focus point and leading line. Based on the values of these three elements, we want to subdivide the canvas into smaller parts according to certain proportions. One of the advantages of generating graphics with software is that objects can be placed and scaled according to exact measurements. This contrasts with our example composition, in which objects were placed on intuition. We have chosen to apply the golden ratio, because it is naturally recursive. This approach can be seen as the inverse of the decomposition of the still life in figure 2.4. The canvas is split up

[1] For example Adobe Illustrator or Macromedia Flash.

[2] Notice that the 'decomposition' in this case is taken rather literally, i.e. we are breaking down the compositional structure.



(a) focus point and leading line

(b) first slicing step

(c) new coordinates

(d) filling with shape pattern

(e) second slicing step

(f) filling of sub-slice: next layer

Figure 3.2: steps in generation of compositional template

into four areas, which we will call *slices*. In figure 3.2(e), one of the slices is divided into four more different sub-slices. The details of this slicing-algorithm are discussed in section 3.

In this section we took the first steps in translating our domain principles to a more concrete framework for the generation of abstract compositions by defining a structural template. Before we give an in-depth knowledge decomposition of our model, we will first explain how we can extend the framework in such a way that it incorporates the rest of the principles of practical graphics theory (section 1.4).

2. Mapping of Practical Domain Concepts

In section 1.4 we analysed our domain by discussing aesthetics, gestalt theory and ending with the more practical graphics principles in section 1.4. We have already applied the most basic of these to the compositional template of section 1. We will now show how the rest of the ideas about practical graphics theory can be mapped to our model. We will take colour, composition and the four principles of graphic design as starting points.

2.1 Composition

We have discussed several composition types in section 1.4. We have chosen the triangular composition here as a starting point because it has a leading line which is easy to represent. It is also reasonably similar to the diagonal composition and it enables us to incorporate ideas about depth composition in our model. Because the splitting up of the canvas into rectangular areas according to certain proportions is similar to a neo-plasticist approach, our compositional model can even be seen as a geometric one. For example, assigning a black stroke of fixed width to the guides and filling some of the slices with the colours red, yellow, blue would generate a result resembling figure 2.1(g).

Because we have defined our model recursively, each slice can contain any number of sub-slices, dependent on how deep recursion will go. This means that there can always be one or more smaller slices that overlap with a part of the original slice. Interestingly, this creates an automatic layering of objects; the interplay of overlapping shapes, both belonging to the same group and to groups at different layers, can achieve the interposition-effect described earlier in the context of depth composition. Thus, the term ‘depth’ not only applies to recursion, but also to the third dimension.

The idea behind the leading line is that it represents the line in which the important visual elements are placed in the composition, so this also includes the focus point. Because this virtual line is formed by the alignment of several objects, the theoretical principle underlying the leading line is that of closure, as discussed in 1.4.

One of the ways to extend our model is by adding points to the leading line, increasing the complexity of the composition. The triangular and diagonal composition can be seen as simple examples of movement compositions. This is even more so when the points are connected by curves instead of lines, giving the composition a more subtle flow. On the other hand, this would make it difficult to say anything reasonable about depth composition. Technically, adding a second leading line would also be possible, but this does not seem desirable because of the emphasis principle, which advocates a single theme.

2.2 Graphic Design

The four principles of graphic design as we have defined them are emphasis, balance, rhythm, unity. We will discuss each of them in relation to our model here.

The two ways of achieving emphasis are contrast and isolation. Our model incorporates a focus point and leading line as part of the abstract representation of the composition. At the same time, these elements indicate the location and direction of the main, to-be-emphasised, visual objects. Attention should be drawn to the focus point first, after which the eyes should be guided automatically through the rest of the composition by the leading line. Therefore, the focus point, the primary element of the composition, should be most emphasised. We can isolate the focus point spatially from the other elements in the composition. Contrast can be achieved by making use of different kinds of shapes near the focus point compared to the rest of the composition and giving these shapes a colour that stands out from the rest. The same ideas hold for the elements around the leading line, although they should be less salient than the focus point.

How to achieve balance at the compositional level has already been explained in the previous section, but applying balance to smaller parts of the artwork, especially through the figure-ground principle, still remains an important issue. Because we will work with proportioned subareas, slices, we have an obvious candidate for a bounding (back)ground to which we can relate the figure that it contains. Because of recursion, this applies to all depths. We could then specify the ratio between the area of the objects the slice contains and the area of the slice that surrounds these objects. As long as we use convex polygons, calculating the area can be done with formula 2.1[3].

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i) \tag{2.1}$$

where n is the numbers of nodes and (x_n, y_n) is assumed to be the same as (x_1, y_1). More complex formulae are available for non-convex polygons and curved objects. A heuristic method to calculate the approximate area of a non-convex polygon is by calculating its convex hull. This is the minimal convex polygon that encloses its non-convex counterpart[4]. There are several convex hull algorithms that have been developed in the field of geometry. One of these is given in appendix II.

But the most important issue has not been discussed yet: how to choose a sensible ratio between the figure and the ground. The normal golden ratio relates to linear measures, not to the area of surfaces. We will have to find a golden ratio for area. The derivative of the square of the Fibonnaci sequence, as depicted in figure 2.3(b), might be a candidate. On the other hand, a less algorithmic and more knowledge-based approach could also be suitable; rules for deciding the size of the figure could be related to depth or contrast-level.

Rhythm is related to a number of other concepts in graphic design and gestalt theory, including grouping, similarity and repetition; a rhythm in a painting must consist of repeating elements that are similar to each other, thus forming a coherent group. How the elements are similar is up to the artist. This is a fundamental issue though. A flowing rhythm, for example, would consist of a pattern in which the parameters for

[3] Adapted from http://www.mathwords.com/a/area_convex_polygon.htm.

[4] This can be compared to wrapping a rubber band around an object.

shape, colour or relative location of subsequent elements would only vary in small amounts. But with a dynamic rhythm, on the other hand, the parameters would vary between more extreme boundary values with non-continuous intervals. We will define a set of *coherence parameters* about which we can reason. For example, in the case of strict spatial constraints, too much variation in location would not be desirable, while larger variations in colour would not be a problem. The coherence parameters we have chosen are:

- location,
- rotation,
- scale,
- skew and
- colour.

The first four parameters are directly related to the SVG transformations. We will discuss colour in section 2.3.

2.3 Colour Schemes

Although colour plays a significant role in aesthetic perception, we decided not to focus on this aspect of visual art. Therefore, we kept our colour model simple. We have chosen to work with the HSL colouring-scheme, as discussed in section 2.1. HSL consists of a scale for respectively the hue, saturation and the lightness of a certain colour. We have chosen to implement a minimal representation of colours for our system; two HSL-encoded colours are specified.

There are several ways to vary colours. This can be compared to the use of a palette; a set of basic colours can be continuously blended. Theoretically this leads to an infinite number of combinations. On the other hand, the colours can be applied to the canvas in unblended form. The process of painting is then dependent on the set of available basic colours. By default, colours in vector graphics are discrete, which is the reason that this last approach is best suited for our model. We will create a set of colours by interpolating between the two input HSL values in even increments. We will determine the step size on the basis of the recursion depth. This results in the fact that more complex compositions will have more subtle colour variations. There are many options for assigning to graphic entities although this is not the topic of this thesis. Therefore, we decided to assign the colours randomly.

3. Structural Decomposition

In this section, we will make explicit the different levels into which our model can be decomposed. This is done top-down in a stepwise refinement from conceptual ideas about composition and graphics towards the formal, representational aspects. The conceptual ideas will eventually be implemented by the back-end, which is concerned with high-level decisions about the composition. A formal representation is important to be able to translate these ideas to vector graphics.

3.1 Compositional Level

The representation of the composition consists of the elements canvas, leading line and focus point. Both in SVG as well as in our own model, every object is a leaf in the tree hierarchy with the canvas as the root. In SVG, these leaves consist of different types of vector graphics (or even bitmap images), key concepts in our model are slices, patterns and shapes.

The leading line consists of several points, connected by lines. Exactly one of these points is the focus point, while two other points are located on one of the edges of the canvas. This way, the leading line seems to 'leave' the canvas. Technically speaking, the triangle of the triangular composition lacks one of the three sides. That is why we prefer to speak of an *angular* composition instead[5]. We have chosen for this representation because we want to treat the leading line as a purely conceptual element. Just like a guide, it has no definitive end, so it cannot stop in the middle of the canvas. The difference is that guides can only consist of single straight lines.

The skeletal framework described in the previous sections is the starting point for the generation of the total composition. From it, the canvas is subdivided into smaller sections using a recursive slicing-algorithm, which eventually results in a mosaic of sub-slices around the leading line. Each step of the recursion results in a slightly more complex and granular scheme of child slices that can be superimposed on the previous layer of parent slices. The result can be seen in figure 3.3, which depicts the stages of the framework at several iteration-levels. The stacking of layers of slices is a fundamental aspect of the depth composition in the system; a calculated parent slice is not thrown away after the sub-slices have been calculated. Instead, it is used as a enclosing border for a shape pattern that is exactly one level deeper. This pattern is then one layer behind the shape pattern of the sub-slice. In the following paragraphs, we will handle some of the important aspects of the algorithm.

The slicing process can be seen as the traversal of a search-tree. A slice is represented as the values for the x and y position of the top left corner, in correspondence with the SVG coordinate system, and the values for the height and width. Also, coordinates for the leading line are included. There are three types of possible slices:

1 the slice encloses the focus point,

2 the slice does not contain the focus point but does contain part of the leading line or

3 the slice contains neither: it is empty.

Because of our three-point model of the angular composition, in the first case the leading line will always consist of three points. In the second, the leading line will always be straight. When an empty slice is generated, the recursion is ended. The reason for this is that the actual shapes will eventually be generated partly based on the leading line coordinates of a slice. An empty slice is always a leaf in the tree-hierarchy of slices. Depending on the user-defined recursion-depth, at the deepest level there will also be one leaf-node of the first form and several of the second. The algorithm is split up into two parts, which correspond to the first two cases. We will discuss them in the next paragraphs.

[5] Angular in this sense would also include diagonal and geometric compositions.

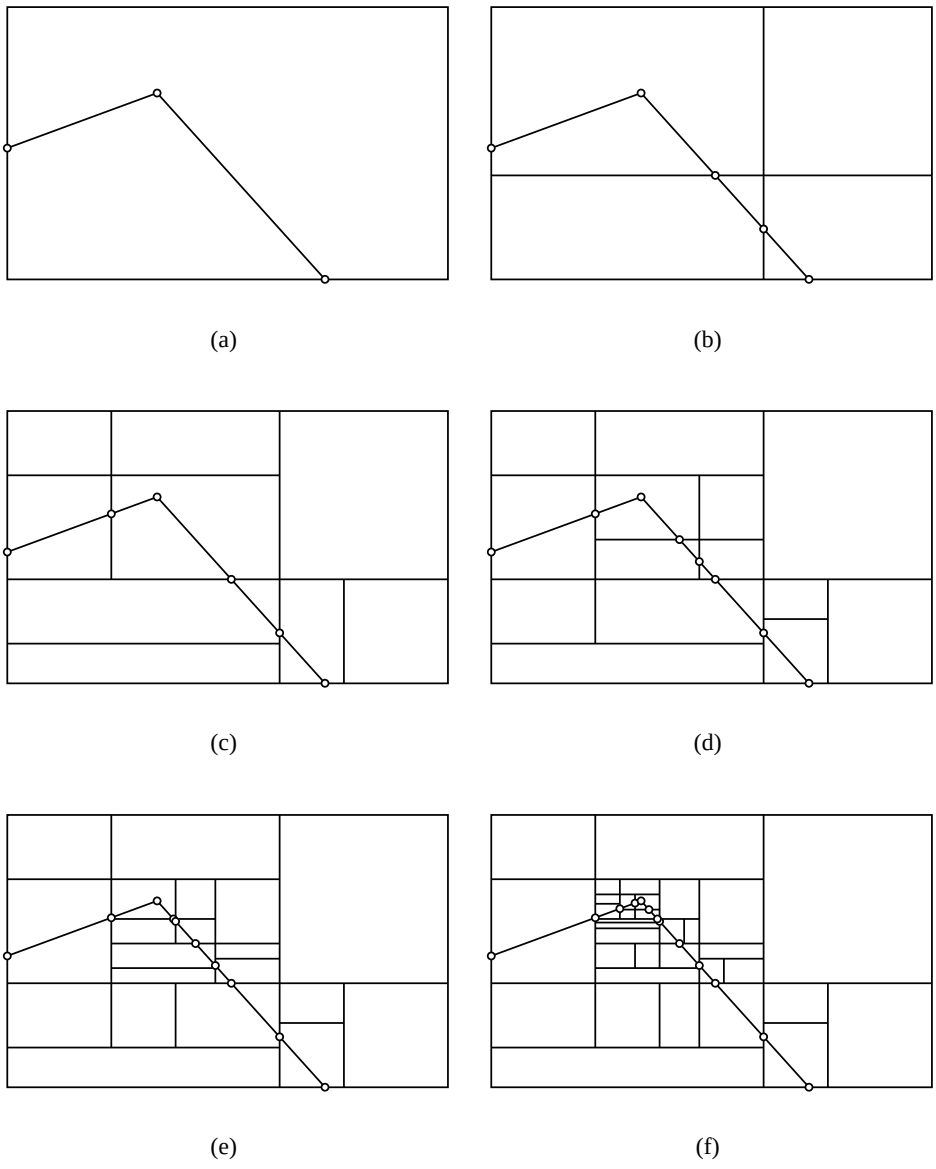
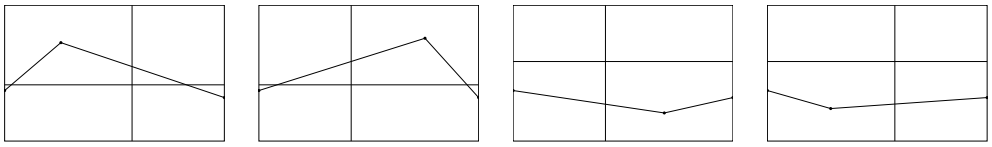


Figure 3.3: first 5 iteration steps in framework generations

The root node corresponds to the initial state of the framework as we discussed in section 1. It represents the entire canvas, including leading line and focus point, so it can be classified as an example of the first case. A rectangular area can be divided into golden ratio proportions in four ways. We have chosen to divide the canvas in such a way that the slice with the largest area corresponds to the location of the focus point. The rationale behind this is that propagation of the slicing algorithm will be more likely the larger the slice is, resulting in a more complex pattern of slices in the focus area. This is for a large part dependent on the stop-condition of the recursion in the second and third case, which will be explained shortly. Termination of the recursion in the first case only takes place when a user-defined iteration-value has been reached. For more details, see appendix III. No analytical evaluation of the likelihood of propagation has been done, however. We will return to this in the discussion of chapter 5.

Assigning the largest slice to the focus point area depends on which quarter of the canvas the focus point is located in[6]. This is depicted in figure 3.4. Slicing of the



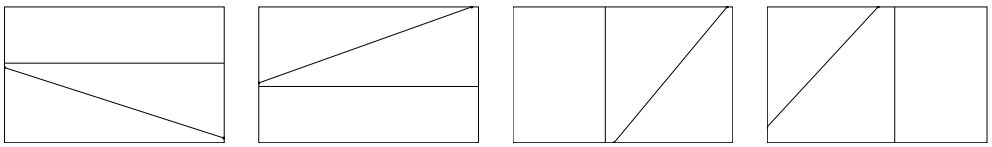
(a) (b) (c) (d)

Figure 3.4: 4 different slicing options

canvas will result in the generation of four new child slices. When the leading line is cross-sectioned by one of the edges of a slice (this is often the case at several points at the same time), the intersection point is calculated. After that, the original leading line is split up into several sub-lines, Each of which is assigned to the corresponding (sub)slice. Exactly one of these will enclose the focus point. The other three will either contain a part of the leading line or will be empty.

Because a slice that only contains a part of the leading line (and no focus point) has a less important place in the composition, the way we will divide it will be simpler. Instead of four, we will now split the parent slice up into two child slices according to the golden proportions. There are four options for doing this: two in the horizontal direction (left or right) and two in the vertical (top and bottom). This is depicted in figure 3.5. How the parent slice is subdivided influences the placing of the leading line in the new slices. This influences the generation of graphics, which is partly based on the leading line coordinates. We therefore decided to let the algorithm try to minimally contain the child slice. The algorithm only succeeds if the leading line is not intersected, otherwise the predicate fails and recursion stops. If it does succeed,

[6]It should be noted that this algorithm is completely dependent on the representation of the leading line as discussed in section 1. When nodes are added to the leading line or curves are applied, a different kind of algorithm has to be implemented. Because of limited time constraints we have decided to only implement the simpler angular composition. However, because of the modular decomposition of our framework, it would be possible to design a new leading line and slicing algorithm without having to re-implement the rest of the system.



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

Figure 3.5: Four options for dividing a slice that contains a straight leading line section

one of the child slices will contain the leading line, while the other will be empty and is therefore discarded.

To summarise, in this section we have discussed several conceptual decisions concerning the generation of the skeletal structure of our compositions. We chose to apply an angular compositional style because it is basic, yet general. The composition is divided according to an algorithm that produces focus point slices, containing the focus point plus an angled part of the leading line, as well as linear slices, containing a straight part of the leading line. This algorithm is dependent on the representation of the composition.

3.2 Graphic Entity Level

The next step is the generation of the actual visual content. As we have explained, the initial representation of the canvas, leading line and focus point, is used to calculate the coordinates of the slices. In our model, each slice essentially represents a graphic entity. This entity can consist of a single shape or composite shapes (a pattern). In this section, we will discuss how shapes and patterns of shapes can be generated based on the slice coordinates.

Because our research is mainly focused on the composition of graphics, we will not define algorithms that generate the actual atomic shape objects. We will discuss several existing shape generation algorithms and how they can be combined with our work in the conclusion of section 5. Instead, we will experiment with several predefined shape sets. The first one is the one applied in the example at the start of this chapter, consisting of elementary convex polygons: triangles, squares, pentagons and hexagons. Because of convexity, the area of these polygons can easily be calculated through formula 2.1. Although this set is reasonably meaningless from an artistic point of view, keeping the number of variables low allows us to analyse the resulting compositions more easily. In addition, we will make use of several other, wilder shape set. Because we view each set as a distinct visual vocabulary, we will not combine them but apply them one at a time. We further discuss the different shape sets in chapter 4.

As was pointed out in section 1.4, two related aspects that are important for placing an object in a composition are the proportions and the alignments. The skeletal structure is built up according to the golden ratio proportions. Therefore, if we align the objects to the edges of the slices, the alignments will perfectly correspond to the golden rule measurements. Suppose we have two objects, a pentagon and a hexagon. We want to align the hexagon, which is the smaller of the two, to the pentagon. To be able to reason about the alignments of shapes, we will define a virtual rectangular area that is wrapped around an arbitrary shape. This we will call the *bounding box*. Fig-

ure 3.6(a) depicts both objects with their corresponding bounding boxes. Alignment of a single side can be done four ways: top, bottom, left and right. Figure 3.6(b) depicts a bottom alignment. Note that this is a relative operation: shifting the hexagon downwards or shifting the pentagon upwards yields the same result. Because alignments are important from a graphic design perspective, we would like to maximise their number. Creating a second alignment can be done by shifting the smaller shape to one of the corners of the larger one. If the hexagon would be shifted to the right corner, two sides of the bounding boxes would overlap of one (figure 3.6(c)). One more side could be aligned if one of the objects is re-scaled (this is also a relative operation). If the aspect ratios of two different objects are the same, scaling would cause all four sides of the bounding boxes to coincide. However, this is often not the case. Possible solutions are changing the aspect ratio of one of the objects to that of the other by stretching it (figure 3.6(f) or central alignment (figure 3.6(e). In some cases, for example with typography, changing the aspect ratio is not desirable.

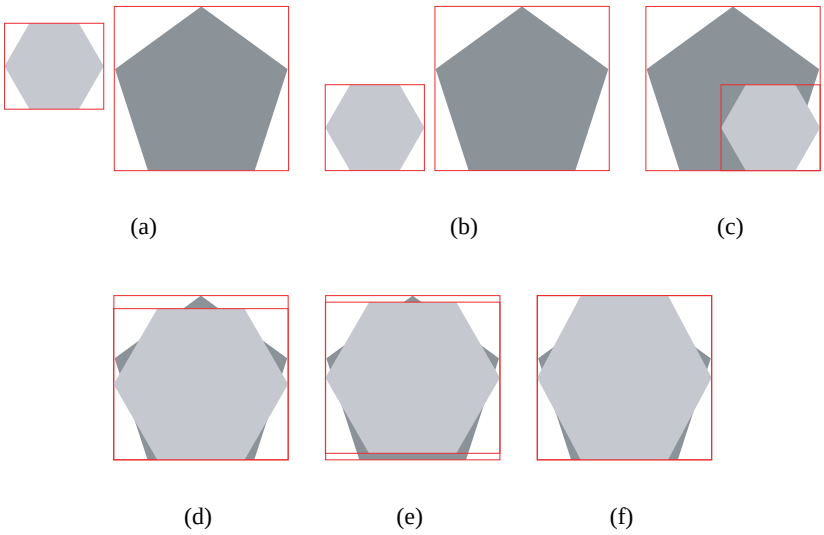


Figure 3.6: several options for alignments

Because in theory well proportioned objects guarantee a certain quality level of the composition, we will try to maximise the number of ‘good’ alignments. If each slice contains a single shape, this is most simply achieved by re-scaling the objects to fit exactly within the containing slice. When a slice contains multiple shapes, additional ideas about groups are required. This is where shape patterns, which we discussed in chapter 2 become important. Shape patterns combine predicate calculus with spatial properties. Therefore, they provide a suitable layer of abstraction between the top-level composition (the skeletal structure) and the visual vocabulary of the shape sets. We will try to incorporate ideas developed by the shape grammar group into our representation.

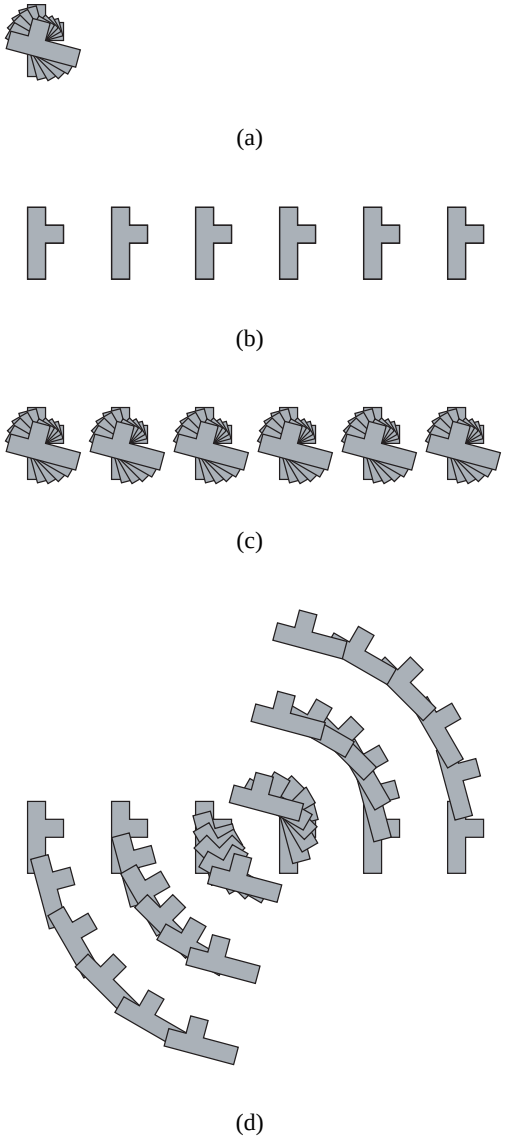
Because shape patterns were originally developed in the field of architecture, there are several differences between shape patterns as described in the literature and the way patterns are applied in our model. First of all, building materials have physical

constraints. We are only constrained to the graphics the screen can display and to the way we are able to represent these graphics with the available technology. Secondly, a building has functional constraints; it is used to live in. This functional constraint can be translated to our goal to design a 'well-formed' composition, depending on the formal principles of composition and graphic design. These ideas are discussed in this chapter. Thirdly, when architecture is seen as a form of applied art, there is a level at which the building conveys an artistic idea. The architect in this case is also a creative artist. The fact that buildings are styled differently can not only be attributed to the combination of different building materials, requirements and constraints. There are just as many architectural styles as there are art movements. In fact, historically, the developments of fine arts and architecture often go hand in hand. This division between a functional, well thought out structure and an actual work of art becomes even more apparent in the domain of generative art. Although it is relatively easy to generate visual contents based on a finite set of shapes and rules, to call something art requires some higher level of understanding. In section 1.3, we already discussed some general philosophical approaches to this issue. In chapter 4, we will discuss to what level we can create 'genuine' art by generation. During the structural decomposition of our framework, we hold on to the more formal approach. Our representation of shape patterns will therefore be based more on de facto combinations of shapes and transformations, with the general background theory of chapter 2 in mind, especially concerning grouping and rhythm.

Because we implement our model in SVG, we make use of the four different transformations: translation, rotation, skewing and scaling. Transformation of a single object or a group is done relative to a certain reference point. We will call this the *anchor point*. In SVG, when a rotation is performed on a shape it has its anchor point on the origin. As a result, it will rotate around its axis. When the shape is translated before it is rotated, the anchor point will stay on the origin. Thus, the location of the anchor point can have a significant effect on the way the pattern is formed. Applying multiple different types of transformations to a single shape can already result in a wide range of distinct patterns. For example, in figure 3.7(a), a figure is translated several times, while in figure 3.7(b) it is rotated around its axis. In figure 3.7(c) the rotated shapes of figure 3.7(b) are translated, while in figure 3.7(d) the translated shapes are rotated. This results in a different pattern structure and shape distribution.

To be able to reason about the alignment of a group, we need to recalculate the bounding box by finding the outer coordinates of the shapes within the group. There is a tradeoff between the rhythm created by the pattern and the alignments of the shapes. Shapes can be aligned to other shapes within the pattern or to graphic entities outside the pattern. Another property that can contribute to a balanced pattern is the distribution of whitespace between the shapes. Because we regard a pattern as a distinct graphic entity, in our model the spatial relationships within the pattern are to some level independent from the spatial relationships of the entire composition. Because the shapes the pattern consists of are at a lower conceptual level, we give a higher priority to the alignment of the entire pattern.

There are several methods to determine the number of shapes used for a single pattern. One is to base the number of shapes in a pattern on the recursion depth. Another alternative is to take a non-linear approach. The exponential and logarithmic functions are interesting candidates, because they are mathematically related to the



(a)

(b)

(c)

(d)

Figure 3.7: switching the transformation order (adapted from [11])

golden ratio. Additional shape pattern characteristics such as direction and center point can be based on the leading line and the focus point. In the case of a linear slice, we will interpolate between the coordinates of the two intersection points. The shapes can then be evenly distributed along the width and / or height of the leading line section that is contained by the slice outline. In the case of a focus point slice, a pattern can be rotated around the center point. We can also define a more complicated distribution based on the center point and the two leading line section.

3.3 Graph Level

To generate output, we need to translate the shape level representation to the level of the output syntax, SVG. In figure 3.8, we have arranged the different SVG classes into a hierarchical diagram, based on [37]. At the top level, a canvas object forms the root of the tree hierarchy. Although we are able to do a scale operation at the canvas level, we will only make use of the width, height and colour[7]. On the canvas, graphic elements can be places which can be nested in groups. Each group can contain an arbitrary composition of different shapes. The transformation operations transform, scale, skew and rotate can all be specified in one group. Groups can also be nested themselves, resulting in more complex arrangements.

A graphic element consists of colour values for the fill and the stroke and the stroke width. Initially, to keep the number of parameters low, we will not use strokes on our shapes. A list of coordinates is specified that defines the actual points in the shape graph. How the lines between these points are drawn depends on the choice of basic shape or path-based shape. In the first case, straight lines are drawn between each point. Because all shapes out of the simple set are polygons, we can use the polygon class of the basic shapes to generate them. We can simply implement the point list as a Prolog list that contains coordinate pairs. If we make use of convex polygons, spatial properties can easily be calculated based on the coordinates. On the other hand, if we want to define more extravagant shapes this would not be the case. Calculation of the bounding box and the shape area would be more difficult. A possible solution could be to predefine a bounding box and a convex hull around the object. An explanation of how a convex hull can be calculated is given in appendix II.

4. Summary

In chapter 2 we analysed the domain of art in general and discussed more formal principles of the domain theory. In this chapter we tried to formulate a general model for the generation of abstract geometric art. We did this by defining a simple method to generate the top-level composition structure and mapping our domain concepts to it. Finally, we took a more formal approach by giving a knowledge decomposition of the model. In the next chapter we will show how we implemented our model and in what type of compositions this resulted. After this, we will discuss to what extent our model is suitable to express higher-level concepts of art and creativity with.

[7] Scaling the canvas is useful when nesting different SVG objects. However, we will only generate a single canvas, so this functionality is not needed.

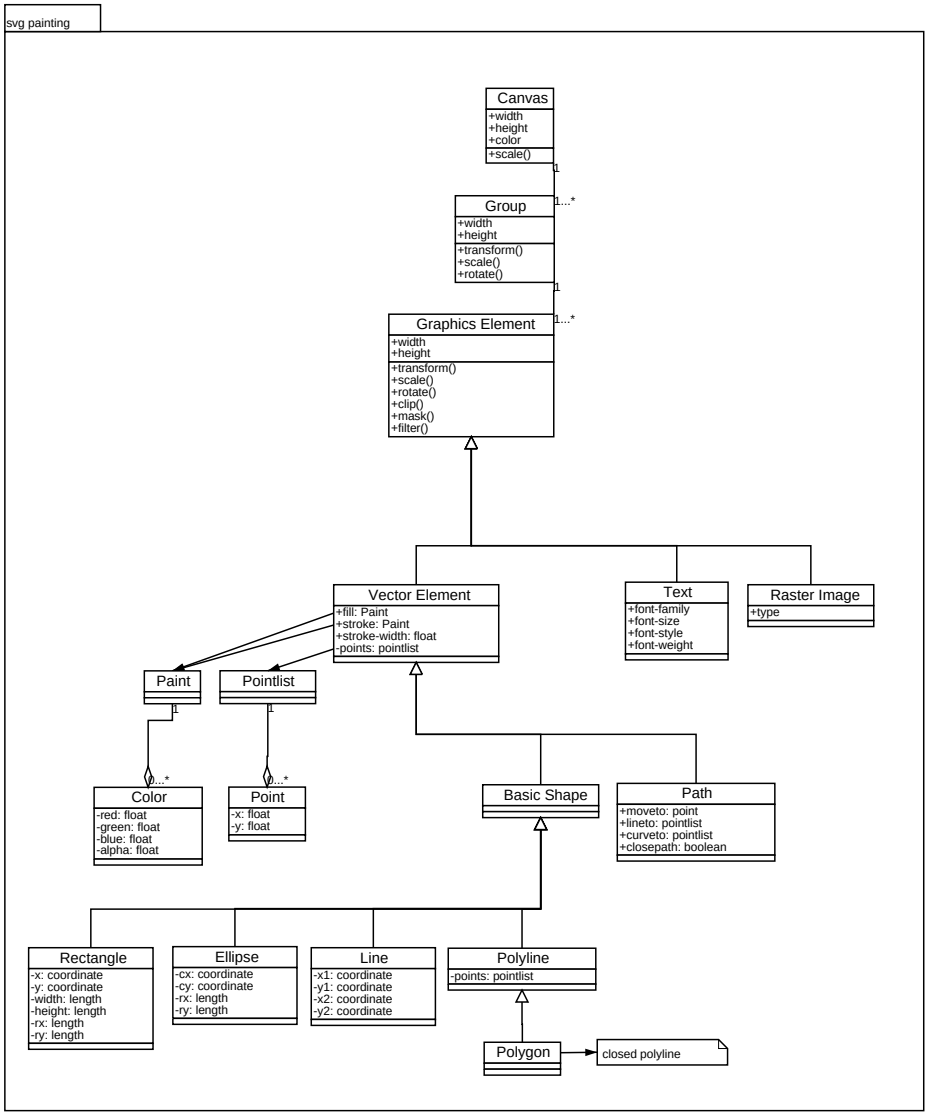


Figure 3.8: UML class hierarchy of SVG

Chapter 4
Results

Based on the conceptual framework developed the previous chapter, we implemented a prototype system. The idea of this system was not to implement a single composition, but to be able to generate many different compositions, combining sound theoretical principles with more specific ideas and randomness. Taking this approach, we were able to produce a large collection of graphics that have a similar style but differ in the way they are arranged. For our project, emphasis lay on research. Therefore, rather than following a strict software design procedure, the system has been implemented while developing most of the ideas contained in this thesis. This has helped us understand many of the conceptual dependencies and practical constraints involved.

We will first discuss the main input parameters and data structures in section 1. After this, we will discuss the implementation of the skeletal composition structure. In section 3 we will discuss how we defined several shape sets that are used as the visual vocabulary to fill the slices in the composition structure and in section 4 we will discuss what problems we faced to implement the more higher-level concepts. Finally, in section 5, we will see how significant the results are from an art perspective. The exact details of the implementation are explained in appendix III.

1. Input Parameters and Data Structures

We will start off with the initial information state of the system. Here is a list of fields that are required for the demo to run:

- the filling mode,
- the shape set,
- the canvas dimensions,
- the leading line coordinates,
- the colour schemes,

- a list of booleans for the rendering the guides,
- the values for rendering the guides and
- the depth of recursion.

The first field is the *filling mode*, which describes the method of content generation. By this we mean the way shapes are aligned, composed and scaled within the slices. Currently, a limited number of different mode options have been implemented, mainly specifying the way shapes are aligned.

The *shape set* is a set of four to five shapes. These shapes have been predefined and can have arbitrarily different characteristics, although we have selected the sets based on certain similarities to guarantee a level of consistency in the visual appearance of the generated compositions. Because the shape sets and the filling modes have been defined independently, we can experiment with different combinations of visual styles and generation strategies.

The *canvas dimensions* simply consist of a value for the width and a value for the height. Preferably, width and height should be chosen according to the golden ratio. Three pairs of *leading line coordinates* are required; one for the left edge of the canvas, one for the right and one for the focus point. For simplicity, we chose not to let the leading line 'leave' the canvas via the top or bottom edge of the canvas. This is done by always setting the x-coordinate of the left leading line point to zero and the y-coordinate equal to the canvas width. This representation guarantees that the leftmost point of the leading line is indeed to the left of the focus point and the rightmost point to the right. This choice has the disadvantage that some angles of the angular compositions are not possible. However, a situation in which the leading line leaves the slice via the top or bottom edge does sometimes occur with child slices. At the moment, this case is not matched by the slicing algorithm, causing recursion to stop.

We have implemented a system which interpolates between two colour values, both of which are encoded as HSL values. This means that two *colour schemes*, each a set of a value for the hue, saturation and lightness respectively, should be provided. The first colour determines the background colour, while the second scheme determines the colour it is mixed with. We have used the colour-conversion libraries of the Cupers Colorpicker System to convert the colour values to the different types of encodings (HSL, RGB and hexadecimal). By mixing the two colours, a simple palette is generated. The number of discrete colours within this palette is based on the recursion depth. Each graphic entity is assigned a single colour from the palette. This means that the shapes contained within a group entity will all have the same, discrete colour. In the current implementation the colours are assigned randomly. To model a sophisticated colour assigning algorithm, more research on this topic would be needed.

In addition to the above, we have added some options for the visualisation of the structure and the debugging of the system. The list of booleans specifies if the edges of the slices, the leading line and the shape bounding boxes are rendered. If so, the values for stroke-colour and stroke-width specified in the next field are used.

We have provided a test file which instantiates the main predicate with some default values. For instance, the canvas dimensions are set to 1000 by 618 pixels for most of the generated example compositions. The guides are rendered in blue with a stroke-

width of one SVG-unit[1]. We have provided several values for the leading line, because this determines the overall composition to a large extent. The values for filling mode, shape sets and recursion depth are left open because these are the most interesting fields. Therefore, these values are varied regularly when testing the system.

2. The Skeletal Composition Structure

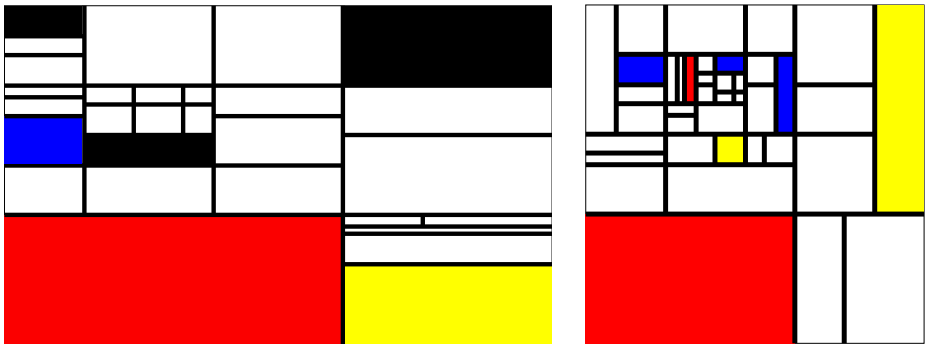
Based on the canvas dimensions, the leading line coordinates and the depth value, we were able to recursively generate the skeletal structure of the composition according to the algorithm described in the previous chapter. After the initial state of the system has been instantiated correctly, the slicing predicate is called. By checking the length of the leading line we decided if the values should be propagated to the *focus point slice* or *linear slice* predicates. In the first step, the leading line contains the two edge points as well as the focus point so the focus point area algorithm will be called. This results in one to four new slices. The new dimensions of the slice and the coordinates of the top-left corner, which provides the anchor point, are calculated. Also, the coordinates of the points where the leading line leaves the new slice are calculated. Thus, at least one new focus point slice and zero to three linear slices, which have a leading line with only two points, are generated. The values of these fields are fed back into the main recursion loop. In its current state, the slicing algorithm is still incomplete. This is due to the fact that at deeper levels exceptional situations will occur within the slicing structure, which our demo does not yet handle, although these situations are rare.

To test our slicing algorithm, in the first stage of our system, we generated some compositions in the style of Mondrian's lattice compositions (see figure 2.1(g)). A simple colour scheme was implemented, which consisted of the primary colours red, blue and yellow in addition to black and white. Note that the colour representation as described in section 2.3 has not been used for this; the set, consisting of the primary colours plus black and white, was predefined in a separate list. For every slice, a single rectangle was generated to which a colour from the set was assigned. We experimented with different probabilities, where the probability for the primary colours and black were set low and the probability for a white rectangle was set high. The proportional guides were rendered according to a stroke width of 4 and the colour black. We experimented with several different settings of the leading line and canvas dimensions. This resulted in the compositions of figure 4.1(a) and 4.1(b).

3. Alignment: Single Shapes

As we mentioned in chapter 3, we made use of several different shape sets. The first one consists of simple convex polygons. We included all the shapes between triangles and octagons. This is depicted in figure 4.2(a). The second set consists of isometric shapes, restricted to angles that are multiples of 45 and 90 degrees. These shapes are depicted in figure 4.2(b). Next to these relatively basic shape sets, we defined two additional sets that consist of more freestyle shapes. The first one, depicted in figure 4.2(c), still includes horizontal and vertical lines that coincide with their bounding boxes. The idea behind this is that the alignment with the slice will still be clear. Another reason is that this way the shape will sometimes merge with a neighbouring shape with (approximately) the same colour. Finally, we defined a set which consists of shapes that

[1] By default, this is equal to one pixel.



(a) (b)

Figure 4.1: generated Mondrian-like compositions

are so angled and curved that they only coincide with their bounding boxes at single intersection points.

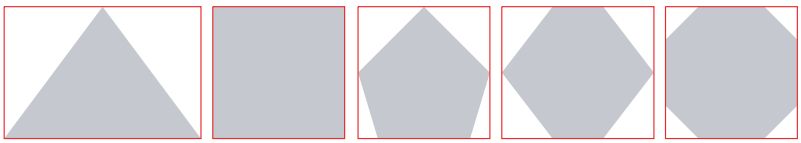
To test the effect of different alignment strategies, we implemented three simple filling algorithms. For each of these, a shape is randomly selected from the current shape set and assigned to a slice. This way, each slice contains a single slice. The first filling algorithm re-scales the shape object from the shapes set to either the height or the width of the slice, depending on the aspect ratios. It then simply aligns the shape to the top-left corner of the slice. This is depicted in figure 4.3(a). The second algorithm randomly aligns the shape to one of the four corners of the slice. This way, the shapes are placed within the slice less deterministically [2]. The result is depicted in figure 4.3(b). The final alignment algorithm also randomly rotates the objects within the slice over angles that are multiples of 90 degrees. This is depicted in figure 4.3(c).

Rotating the shapes to angles that were not multiples of 90 degrees turned out to be more difficult. The reason for this is that the bounding boxes of the objects needed to be recalculated. If this could be done by simply rotating the bounding box and finding the new outer coordinates with a goniometric function, there would be no problem. However, for most shapes, rotation causes the horizontal and vertical outer points to change. As a result, if we want to reset the bounding box of an arbitrary shape, we have to analyse the entire outline of the shape. This is illustrated in figure 4.4.

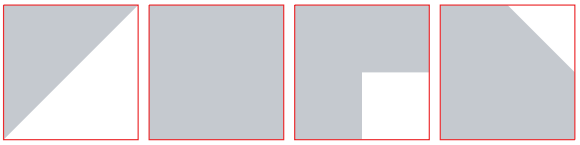
The result of the filling of the composition structure with the polygon shape set can be seen in figure 4.3(d). We have chosen colours with HSL values that lie far apart to emphasise the separate objects. Based on this simple setup, some observations can be made. First of all, many of the objects are aligned to each other, which was the main goal of the simple shape algorithm. However, because the shapes are so basic, they are instantly recognizable as polygons. As a result, there is a certain level of balance, but a higher sense of unity is missing.

To reach more unity in the composition, we replaced the polygons with the isometric

[2] Because the shape is scaled to fit exactly within the slice to maximise the number of alignments, either the horizontal or the vertical direction will become unimportant. As a result, in practice there will only be two alignment options: either top and bottom or left and right.



(a) polygons

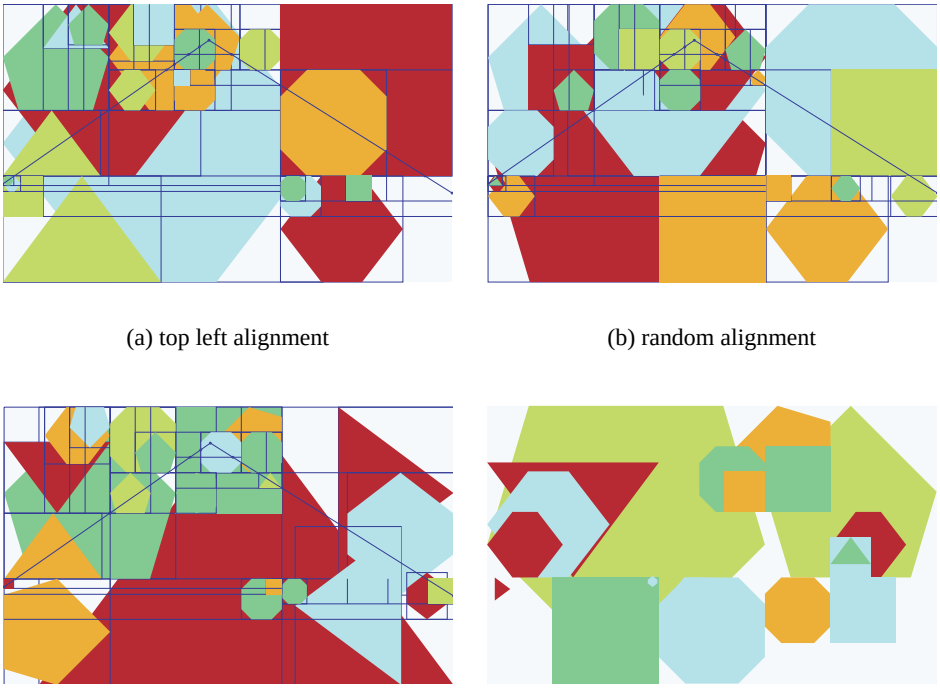


(b) isometric

(c) freestyle set 1

(d) freestyle set 2

Figure 4.2: shape sets



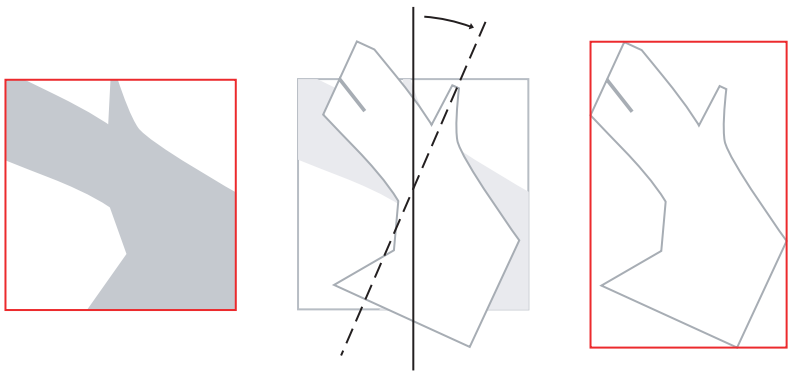
(a) top left alignment

(b) random alignment

(c) random alignment plus rotation

(d) result without guides

Figure 4.3: generation based on polygons



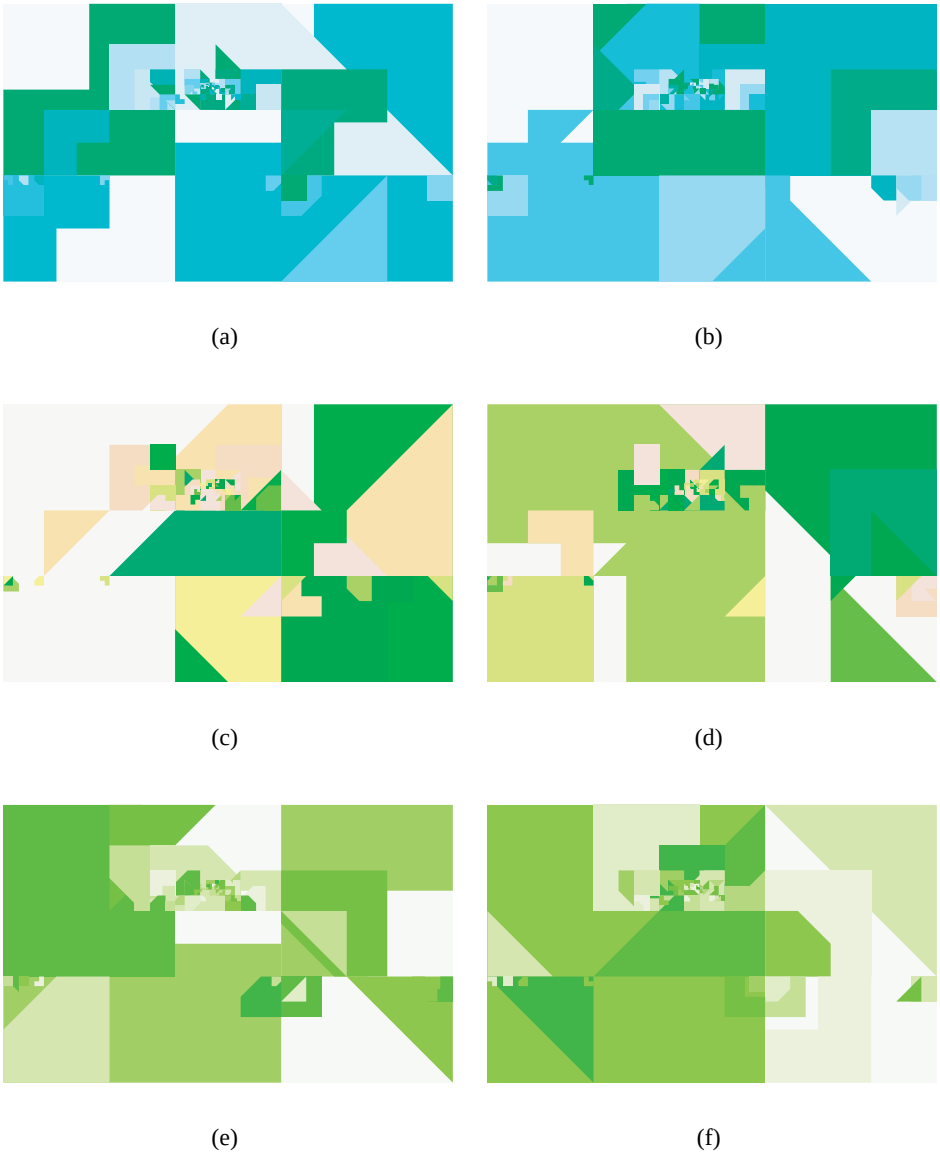
(a)

(b)

(c)

Figure 4.4: recalculating the bounding box

shape set. This set has some interesting properties. First of all, because the angles of the objects within the set are restricted, so are the angles in the entire composition. Secondly, the triangle and square are sub-shapes of the other two shapes. And finally, when the objects are placed within the framework, the objects will seem to fall together like a puzzle. This is caused by the isometric angles and the coinciding of parts of the shape outlines with the edges of their bounding box. For this example, we chose colour values that lay closer to each other, to enhance this shape fitting effect. A disadvantage of this is that the simplicity of the colour model becomes more apparent. Some of the results are depicted in figure 4.5.



(a) (b)

(c) (d)

(e) (f)

Figure 4.5: generation based on isometric shapes

Finally, we filled the slices with freestyle objects. Because these shapes are more wildly curved and (especially in the second set) the outlines coincide less with the bounding box edges, there is much more ground area. Also, some them are *compound shapes*, which means that they have empty areas enclose within the shape outline. The figure-ground ratio is therefore usually much smaller than that of the shapes in the polygons and isometric sets. This has a significant effect on the balance of the entire composition. We experimented with different colour schemes. The results of filling the composition structure with the freestyle shape sets are depicted in figure 4.6. There are a couple of observations that can be made. First of all, the use of the freestyle shape sets often results in very chaotic compositions. The wild characteristics of the freestyle shapes cause the alignments to be less apparent. Furthermore, there is less balance because of the many different angles in the shapes. Finally, the shapes do not seem to flow into each other as well as with the isometric shape set. On the other hand, whereas with the polygons set the shapes were always recognisable as polygons and with the isometric set all the composed shapes had triangles and squares as sub-shapes, layering the freestyle shapes often results in the emergence of completely new shapes.

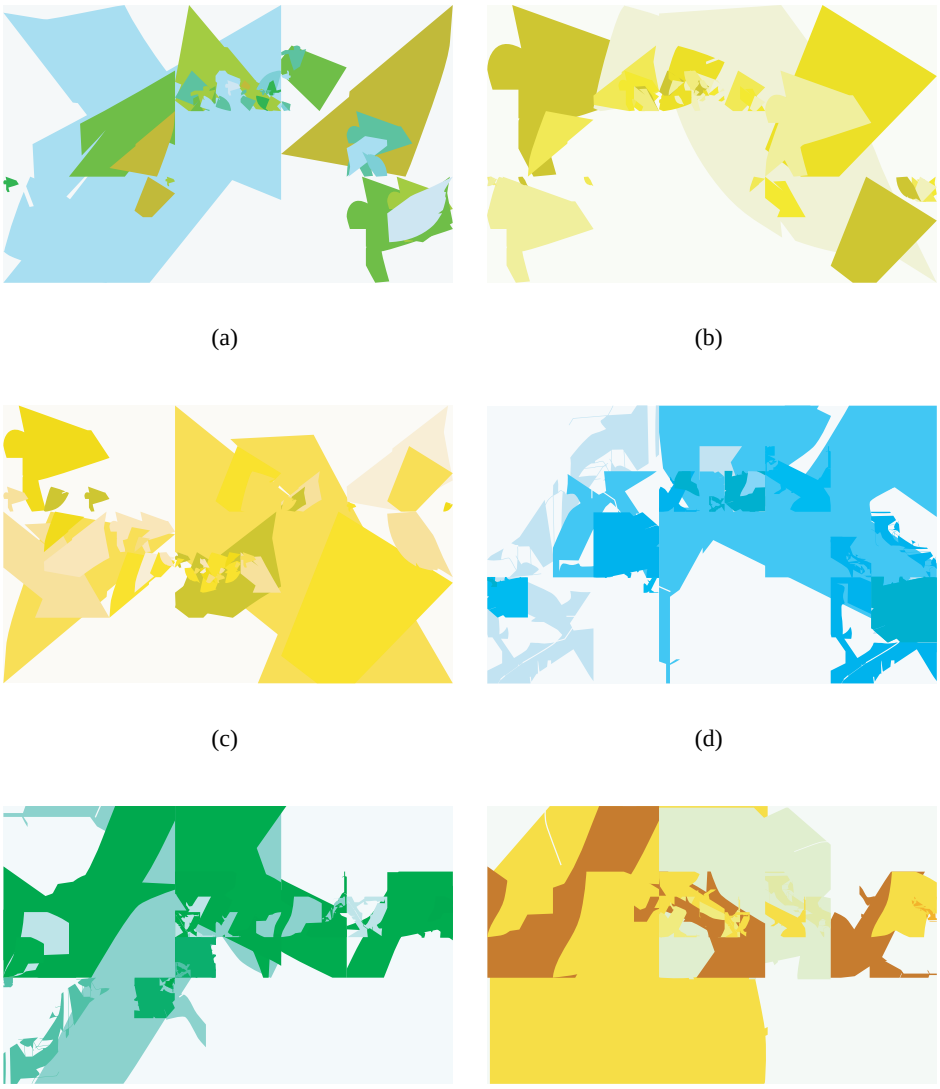
4. Patterns: Multiple Shapes

Several main concepts of our model have been implemented. At the top level, the composition structure includes the angular composition style as well as a layered subdivision of the canvas into well-proportioned slices. For the generation of the actual visual contents we have designed several shape sets to use as a visual vocabulary. We now want add shape patterns to our implementation. As discussed in chapter in chapter 3, the grouping of shapes is important because it adds rhythm to the composition and we are able to represent more complex compositions, filling the gap between the composition level and the shape level of our knowledge decomposition.

For the generation of patterns we encountered the bounding box resetting problem again (see figure 4.4). Because we already did some experiments with alignments, we decided to ignore this problem and generate graphics that did not correspond to the golden rule proportions. As a result, in this case our framework is used only to generate the graphics by and does not guarantee that the shapes are aligned correctly anymore.

The patterns were generated based on the slice dimensions, recursion depth and the leading line coordinates. To keep it simple we used the polygons shape set for this example. The number of shapes used for a pattern was determined based on the current depth value. We implemented two simple shape pattern algorithms, one for the focus point slices and one for the linear slices. For focus point slices, we defined a pattern that rotates shapes around the focus point. The rotation step size was determined by simply dividing 360 degrees by the value of the current recursion depth. The shapes were translated before rotating them. This is comparable to the pattern in figure 3.7(d), only with translations of single shapes. The translation distance was determined by calculating the closest distance between the focus point and a slice edge.

For linear slices, shapes were rotated before they were translated, comparable to figure 3.7(c) but again with single shapes. This resulted in a more linear pattern, in contrast to the circular patterns of the focus point slices. The direction and step size of the translations were determined based on the leading line. The rotation was de-



(a) (b)

(c) (d)

(e) (f)

Figure 4.6: generation based on freestyle shapes

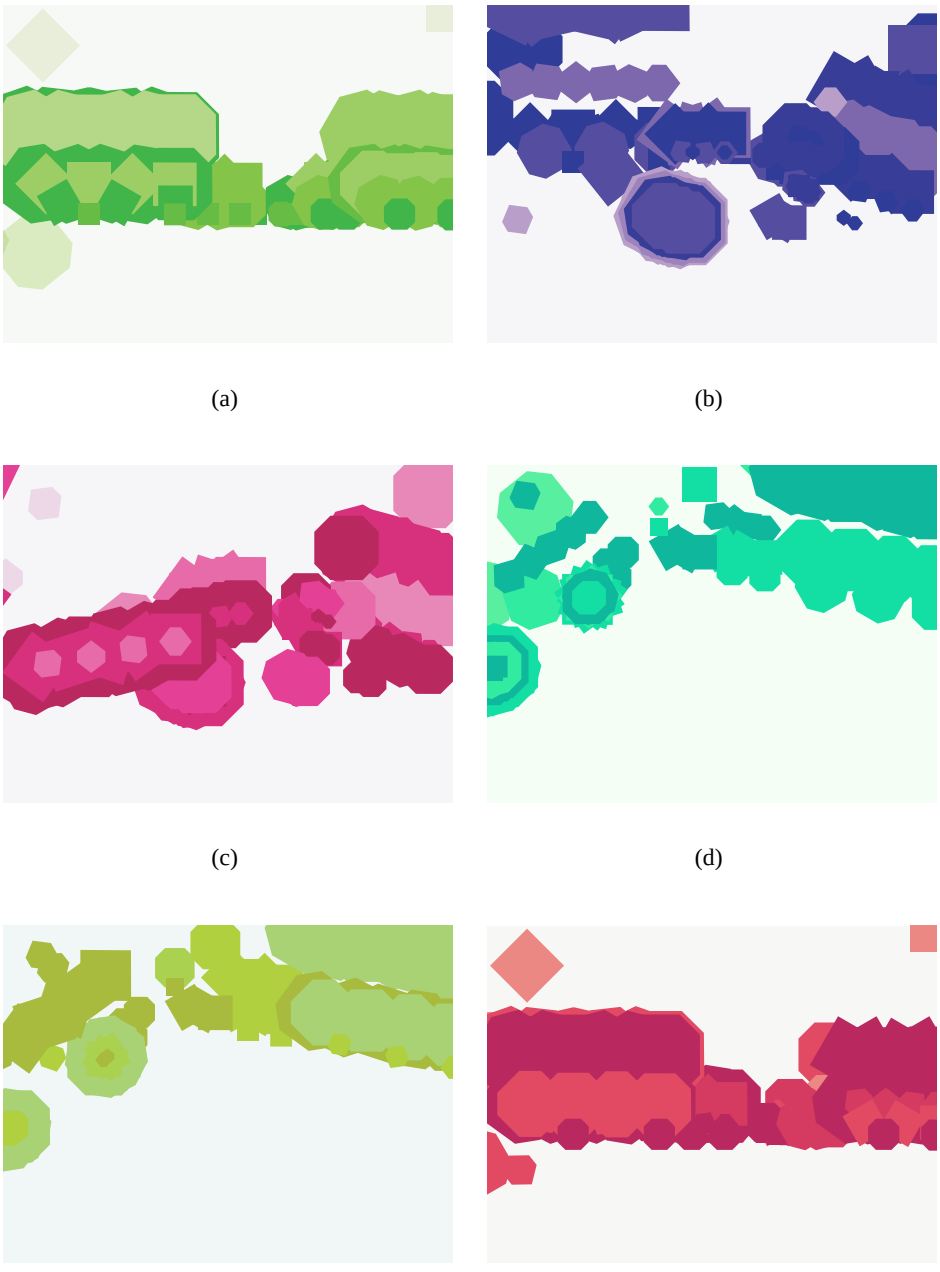
termined by dividing 180 degrees by the value of the recursion depth. In addition to translation and rotation, for the linear slice algorithm we also made use of skewing. A list of skew values was generated randomly. We kept the skew offsets relatively small to achieve a just noticeable distortion of the shapes. Figure 4.7 depicts some of the results.

Although these pattern algorithms implement the ideas of chapter 3 to some extent, they are still very simple. There are several ways we could improve the pattern compositions. First, correct alignments of the patterns could be achieved by ignoring skewing and restricting the rotations to multiples of 90 degrees, just like in the third alignment algorithm of section 3. The next step would be to apply the different alignment strategies to entire shape groups. Also, considering alignments while scaling and translating the shapes within a group would improve the overall balance of the composition. Here, the conceptual relationship between the pattern and the shape(s) it consists of becomes important. We will discuss more intelligent, conceptual level aspects of art generation in the next section.

5. Artistic Significance Results

Up until now, we have taken an analytical approach. Based on the domain theory, we have generated a solid framework. The next step was to generate the visual content in an implementation. We have shown how the ideas described in chapter 3 have been included in it. The application of a general compositional style and graphic design principles guaranteed that from a formal point of view the results were well-formed, although there are many remarks that can be made. At this point, a fundamental issue in the generation of art is encountered: a higher conceptual idea of the artistic content of the generated work is still missing. In this section, we will discuss to what extent we can generate genuine art in generally and specifically with the ideas described in this thesis in mind.

Because this project was done for a masters degree in artificial intelligence, we have tried to scientifically analyse the concepts involved in creating art. This has been a very rational process. However, as we have pointed out in section 3.2, although it is relatively easy to generate visual contents with the computer as a tool, to call the result art requires some higher level idea being conveyed through it. These type of ideas, which give the artwork its artistic meaning and thus its value, are most likely born at a subconscious, irrational level. How to incorporate such ideas in a computer (program) is one of the main points of discussion for AI-based art generation. In an attempt to contribute to this discussion, we already mentioned some approaches to this issue in section 1.3. To summarise, we distinguished three different views. The strong-creativity view states that the computer is autonomous and also consciously aware of its own creative process. The weak-creativity view, on the other hand, does claim that the first aspect, autonomy, is possible, but the second is unimportant as long as the difference cannot be told between the computer generated work and art that has been made by a human. The difference between these two views can be compared to the larger philosophical discussion about artificial intelligence and consciousness [20]. However, in the context of art, the conscious or subconscious development of an idea is essential to the quality of the outcome. Therefore, the distinction between strong and weak AI is not as useful as with more applied domains in which the main requirement is functionality. The last view is that of the programmer as the artist. In



(a) (b)

(c) (d)

(e) (f)

Figure 4.7: implementation of simple pattern-based compositions

this case, the computer can be seen as a very sophisticated tool. There are several levels of complexity at which this tool can be used. We will discuss this in the following paragraphs.

Using a graphics editor could be seen as the most direct way of generating art with a computer. The human user has as much control over his creative output as the software and the human-computer interface allow. In this case, there are not much high-level ideas involved at the side of the computer. There is no significant advantage of using the computer instead of the hand, because generally the results could just as well have been produced with paint and brush. A second step would be to create more time-based output, for example video. This type of content can also be produced a large extent with traditional tools like cameras editing tables, although use of a computer can usually have an added value by speeding up the process.

Using a computer becomes really interesting if we want to add algorithms, adaptive behaviour or even some level of autonomy. These possibilities are hardly available in traditional tools. With algorithmic art, visuals are created based on the artist's description. Application of algorithms in the generation of graphics can be useful for more repetitive and precise tasks, which would require a lot of time and effort of the artist, while the actual task is quite futile. Adaptivity can be achieved by applying machine learning techniques. This approach emphasises the dynamic nature of the medium. Although machine learning is part of the field of artificial intelligence, it does not explicitly represent high-level concepts. It is questionable if compositions that are based on these type of algorithms can be said to have any meaning in them.

Finally, a graphics generating tool could supply autonomous components to facilitate the creative process of the artist. This could be compared to software agents that autonomously help the user achieve his goals, for example finding a certain topic on the internet. Although in practical domains such as this autonomous agents can indeed be very helpful, the question is to what extent the user can benefit from autonomy in a creative environment. How can the software know what the goal of the artist is if this goal cannot be precisely described, changes dynamically and is often not even explicitly known by the artist himself?

In this project, we have chosen to stay close to the meta-art approach. We generate an arbitrary number of compositions based some formal ideas and a certain level of randomness. At this point we have generated a framework and a visual vocabulary, but we still have to develop the language to use this vocabulary. To incorporate higher-level ideas in our model, we could either develop our own creative ideas or do more research on the ideas of other artists. Take for example Mondrian's lattice compositions. Although there are some conceptual differences with original Mondrian compositions (for example, the coloured squares should sometimes align to other lattices instead of to the ones directly containing them) clearly it is relatively easy to approximate Mondrian's visual style with computer graphics. This could be attributed to his extremely minimalist approach to abstract geometric art, which is highly suitable for representation with simple rule-sets. On the other hand, there are also more conceptual theories involved in his paintings. For example, Mondrian would never paint one of the larger areas black, but instead would try to balance the colours to the area, experimenting with the balance of the composition. Kandinsky's paintings often symbolised meta-physical ideas. Many of his compositions are also based on more or less explicit rules, by which conveyed a deeper sense of meaning.

6. Summary

In this chapter we have shown how the ideas of chapter 3 can be implemented to a certain extent. Although our demo is relatively simple it includes most of the basic concepts of our framework. We have discussed the generation of the composition structure, as well as the generation of visual contents using alignments and shape patterns. To analyse to what extent genuine art can be generated in general and specifically for our model, we added a short discussion about the artistic significance of our results. We will conclude this thesis in the next chapter.

Chapter 5
Conclusion

To conclude this thesis, we present a summary of the thesis, present our conclusions on various advantages and disadvantages of the conceptual decisions made during this project (2) and discuss possibilities to extend our model in the future work section (3).

1. Summary

In chapter 2 we have shown that the domain of expressionist art, combined with more formal knowledge about aesthetics, gestalt theory and graphic design, can be used as basis for the generation of abstract geometric compositions by a computer program. After analysing the domain theory, in chapter 3 we designed a conceptual framework. Based on this framework, we have implemented a system to show to what extent we can use this framework to generate abstract geometric compositions. We also discussed the possibilities of including higher-level artistic ideas into generative art systems.

2. Discussion of Conceptual Decisions

For our research we have mainly focused on the formal aspects of composition. We have chosen for the domain of abstract geometric art because the basic principles are formalisable and the visual style can be reproduced with vector graphics. Because we were primarily interested in entire compositions, emphasis lay on concepts such as compositional style, alignment and proportions. However, because one of our goals was to generate compositions based on our analysis, we also had to include lower-level concepts such as shapes and groups into our representation. We decided to keep these as general as possible, describing the role of the graphic entities instead of methods to generate them. For our implementation, we made use of several predefined shape sets. Theoretically, the shapes in these sets could be replaced with ones that are generated. The top-level representation of the composition consists of the three elements canvas, leading line and focus point. We chose for the angular composition style as the starting point because it is both general and simple.

In section 5 we discussed the importance of higher-level artistic ideas behind the generated compositions. This is an issue that is inherent to the domain of art. We have

tried to find out to what extent we can generate art from the formal domain theory. We gave several options for including these kind of ideas into generated artwork. We approached generative art as meta-art, where the programmer had the role of creative artist behind the generated work. Because the main goal of this project was to do scientific research, we did not include our own artistic ideas in the model. On the other hand, neither did we include existing higher-level artistic concepts. The main reason for this is that these kind of ideas are difficult to formalise because it is hard to determine how and why they were formed by the artist. Having said this, it should be possible to include some higher-level ideas into our model by painters such as Mondrian and Kandinsky, who took a relatively formal and explicit approach to painting.

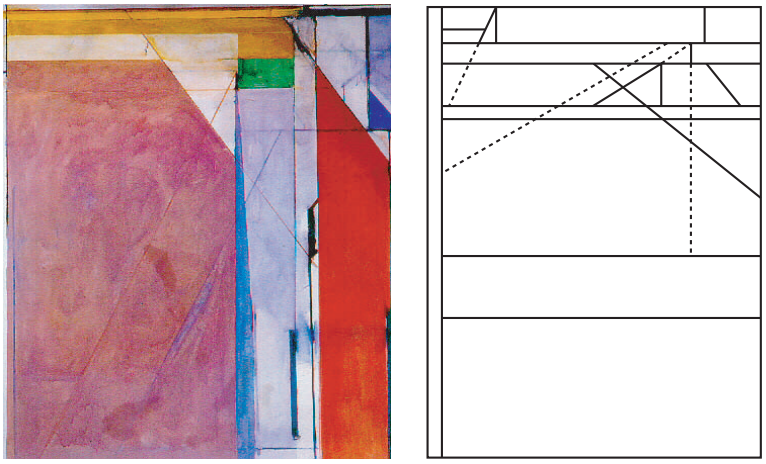
The choice for a recursive definition of the generation process made it easier to implement our demo in Prolog. The framework was generated and at the same time the information of the generated slices was used to generate the visual contents. A more detailed explanation of the implementation is given in appendix III. Proportions are calculated according to the golden ratio. The division of the canvas into slices is done relative to the focus point and leading line. For the filling of the slices with shapes and patterns, first several mathematical operations are done to fit the shapes to the size of the slice, after which the alignments are determined relatively. Thus, our representation combines quantitative and qualitative spatial reasoning.

We have chosen to analyse a very distinct artistic style. Our choice for geometric abstract art made it easier to translate artistic concepts to a formal model. Different art movements are probably more difficult to represent. Post-modernism, for example, is primarily based on the higher-level contents and ideas that our framework lacks. On the other hand, theories about composition and graphic design should also generally apply to more traditional movements.

3. Future Work

There are many ways in which we could extend our model. First of all, the three point representation of the leading line could be changed into a more sophisticated shape, for example a curved line. This would also mean that the slicing algorithm should be redefined, because it is entirely dependent on the way the leading line is represented. It should be interesting to see how different slicing schemes affect the structure of the composition. Statistical analysis of the way the slices are distributed could provide a mathematical basis for comparison. On the other hand, from a more conceptual point of view, it would also be interesting to see how different composition styles, such as symmetric and geometric ones, can be added to our model. Other options for adapting the structure of the composition include the addition of multiple leading lines. In [28], canvases by Diebenkorn are analysed with shape grammars. Based on this analysis new compositions are generated. Besides making use of multiple leading lines (figure 5.1), partial guides (dotted lines) are also represented. This way, non-rectangular slices can be modeled.

Besides extending the structure, including more concepts to achieve emphasis could also help in refining the composition model. As was discussed in chapter 2, the main ideas to increase emphasis are the addition of a focus point, of contrast and of isolation. At the moment our colour model is very simple. This often decreases the aesthetic value of the results. We could define a more sophisticated representation that highlight the objects in the focus point. This would result in the fact that these shapes are isolated



(a) ‘Ocean Park 70’, Diebenkorn

(b) generated compositional structure

Figure 5.1: analysis of Diebenkorn compositions with shape grammars (adapted from [28])

from the other shapes. Also, there would be more contrast between the focus point and the surrounding objects.

Because our research was focused on two-dimensional compositions, we have left issues about depth largely outside the discussion, although the triangular composition seems to be a suitable representation for linear perspective. On the other hand, a realistic depth representation would probably require a much more sophisticated model of graphical objects.

At the graphic entity level, much research has been done by others. We have already mentioned the work of the shape grammar group, on which we have based part of our domain theory. Furthermore, our predefined shape sets could be substituted with other sets with interesting characteristics. For example, similarly to the isometric set, we could define a set of shapes that have angles that are multiples of 30 degrees.

We could go a step further and define shape generation algorithms or incorporate available algorithms into our code. This would open the door for an analysis of the lateral dependencies between shapes, their role within the group or even their role within the entire composition.

Taking technical aspects into consideration results in some new directions we have not yet discussed. There are still many rendering possibilities that are offered by SVG that we have not used. These include changing the stroke characteristics, filling the shapes with (bitmap) textures or gradients (or even leaving the fills of shapes empty) and applying filters to shape objects.

Throughout this thesis, there has been a balance between the analysis of the domain theory and the design of a generation framework. Although our work was not done from a practical point of view, we have designed a general model for the description of composition. It would be interesting to see if this model can be applied to analyse

existing art. This way, collections of artworks could be analysed and classified, which could be an interesting application for musea with online databases.

4. Evaluation Mechanism

In the previous chapters we have described and implemented a model of a system that is able to generate a composition from scratch. We have incorporated all kinds of ideas into the framework, related to composition, design, psychology and aesthetics. Although the generated results are based on these ideas, an evaluation phase could be useful because the contents of the artwork are added incrementally. Therefore, it is difficult to oversee the effect each distinct addition has on the final result. This is the reason why a holistic evaluation mechanism could be desirable. Also, turning generation of the compositions into an iterative process could lead to interesting results: the complexity of the results could be increased and we could research if the results converge or emergent behaviour occurs. It would be interesting to see how we could apply machine learning techniques in this context. Through analysis of the output composition we could refine the input parameters. More extremely, it would even be possible to apply shape recognition techniques to the output. There has already been done extensive research in the area of facial recognition. There also exist pattern recognition algorithms for simpler application areas such as the conversion of images to ASCII-art.

4.1 Quantitative Evaluation

The theory described in chapter 2 provides some good starting points to search for evaluation dimensions. However, although graphic design principles (such as unity, emphasis and balance) could be translated into evaluation criteria heuristically, they are very general concepts that are difficult to quantify. If we want to quantitatively evaluate the generated graphics, we have to find more explicitly definable criteria. The figure-ground principle seems to be a good candidate. It is relatively simple to calculate the area of the shapes. This can be done using formula 2.1 when the shapes consist of convex polygons, or else making use of a convex hull algorithm as a heuristic (see appendix II). The *figure-ground ratio* could then be calculated by dividing the resulting value by the area of the slice.

A next step would be to compare the properties of different shapes to each other. For each slice, graphics are generated independently. As a consequence, graphics that belong to different slices sometimes conflict with one another. The depth order in which patterns are placed plays an important role in this respect. Sometimes a large part of the graphics contained by the parent slice are hidden by those contained by its child slices. We can make a numerical comparison by calculating the figure-ground ratio of the parent slice and the added figure-ground ratio of the child slices. If the ratio of the child slices is larger, we could decide to adapt some of the properties of the composition. There are several options for doing this. First, we could adapt the distribution or scale of one or more graphics. A second option would be to swap the contents of one of the child slice with the contents of the parent slice. Finally, we could swap the shape object the pattern consists of with another, smaller shape.

In the case the slice contains a pattern, some additional remarks can be made. In a pattern, shapes sometimes overlap. This makes it more difficult to calculate the area of the pattern. Research of geometric properties could be useful in this respect. A re-scale operation can be done on the entire group or on the separate shapes within the

pattern. This has a significant effect on the distribution, the figure-ground ratio and the absolute bounding box coordinates.

4.2 Qualitative Evaluation

Although mathematical analysis of geometrical properties can provide a formal basis for evaluation, from an artistic point of view it makes less sense. We could take a more conceptual approach by evaluating relative properties of the generated compositions. First of all, we could consider the alignments. As we explained in chapter 3, there are several options of aligning two objects. Up until now, we have focused on the alignments of single objects within a slice. It makes sense to reason about the alignments of multiple objects within a slice. It would be even better to see how objects of different slices are aligned to each other, because now we would be reasoning about the representation at the composition level. This would be a step towards the expression of high-level ideas about content. Again, we refer to the example of Mondrian, who tried to achieve more balance by placing several objects in counter composition.

It would also be interesting to see how the compositional structure can be adapted (semi-)dynamically. We could take criteria such as emphasis and continuation into account to enhance the skeletal structure. For example, in many of our results the leading line is not clear anymore, leading to chaos. We could try restore its integrity by taking a look at how the objects are relatively distributed. We have already discussed how we could enhance the contrast and isolation of the focus point with colour. We could also choose for a more compositional approach to achieve contrast and isolation by moving objects further away from the focus point, or more to the back. Finally, an interesting option for changing the compositional structure would be to add, merge or remove slices. Although the question is how we could validate such adaptations, if we would implement a sufficiently sophisticated demo interface, we could make changes by hand, turning the evaluation process into a type of supervised learning. Merging a slice could also imply merging the contained graphics. Adding patterns to each other could be done by composing the shapes of the different patterns, and adding the transformations. Here the aspect of transformation order becomes relevant again.

Finally, we could try to evaluate the results based on the semiotics of the contents. This would take us a step closer to genuine artistic content. For example, a more quality based colour theory could support the artistic evaluation of the compositions. For example, we could distinguish between warm and cold colours and try to define a model in which these ideas are linked to the underlying meaning of the results. Some research on colour has been done in [32]. The choice of a more meaningful visual vocabulary would provide us with a way of expressing higher-level concepts.

5. Conclusion

At the start of this thesis we stated the question: ‘How can artificial intelligence techniques be combined with vector graphics to create geometric abstract art?’ We tried to answer this question by both analysing the domain of generative art as well as designing a model to generate abstract compositions. We have shown that it is possible to generate abstract geometric compositions based on ideas of expressionist art movements and more formally, ideas behind the philosophical discipline of exact aesthetics, the psychological gestalt principles and practical theories of composition and graphic design. We then showed how the main aspects of our model can be implemented with

quantitative and qualitative reasoning and how this relates to the creation of genuine art with computer technology. It turned out that our formal research did not provide some of the critical aspects of art. We conclude that although we designed a simple, general framework for generating compositions based on which we can implement a system which incorporates our formal ideas, it lacks the essential artistic concepts that define the quality of the artwork. Although this is a difficult topic, some form of higher-level creativity might be added by either taking a more creative approach, or by doing more research on the ideas of artists that took a explicit, rule-based approach to painting, such as Mondrian and Kandinsky.

Appendix I
Transformation Matrices

Here are the specific matrix values for the SVG transformations as described in section 3.3. This is the translation matrix, where *tx* is the horizontal translation distance ant *ty* is the vertical (adapted from [37]):

$$\begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix} \tag{0.1}$$

For scaling, *sx* and *sy* are multiplication values while the other values are set to zero.

$$\begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{0.2}$$

Rotation makes use of goniometric functions $cos(a)$ and $sin(a)$, where *a* stands for the rotation-angle.

$$\begin{pmatrix} cos(a) & -sin(a) & 0 \\ sin(a) & cos(a) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{0.3}$$

Skewing is done by making us of the $tan(a)$ function; there are two variations, skewing in the vertical and skewing in the horizontal direction. In the horizontal case, the transformation matrix looks like this:

$$\begin{pmatrix} 1 & tan(a) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{0.4}$$

In the vertical case, the $tan(a)$ function is placed differently:

$$\begin{pmatrix} 1 & 0 & 0 \\ tan(a) & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{0.5}$$

Appendix II
Convex Hull Algorithm

This is an example of how the convex hull of a shape is calculated from its set of points using the *Graham scan algorithm*[1].

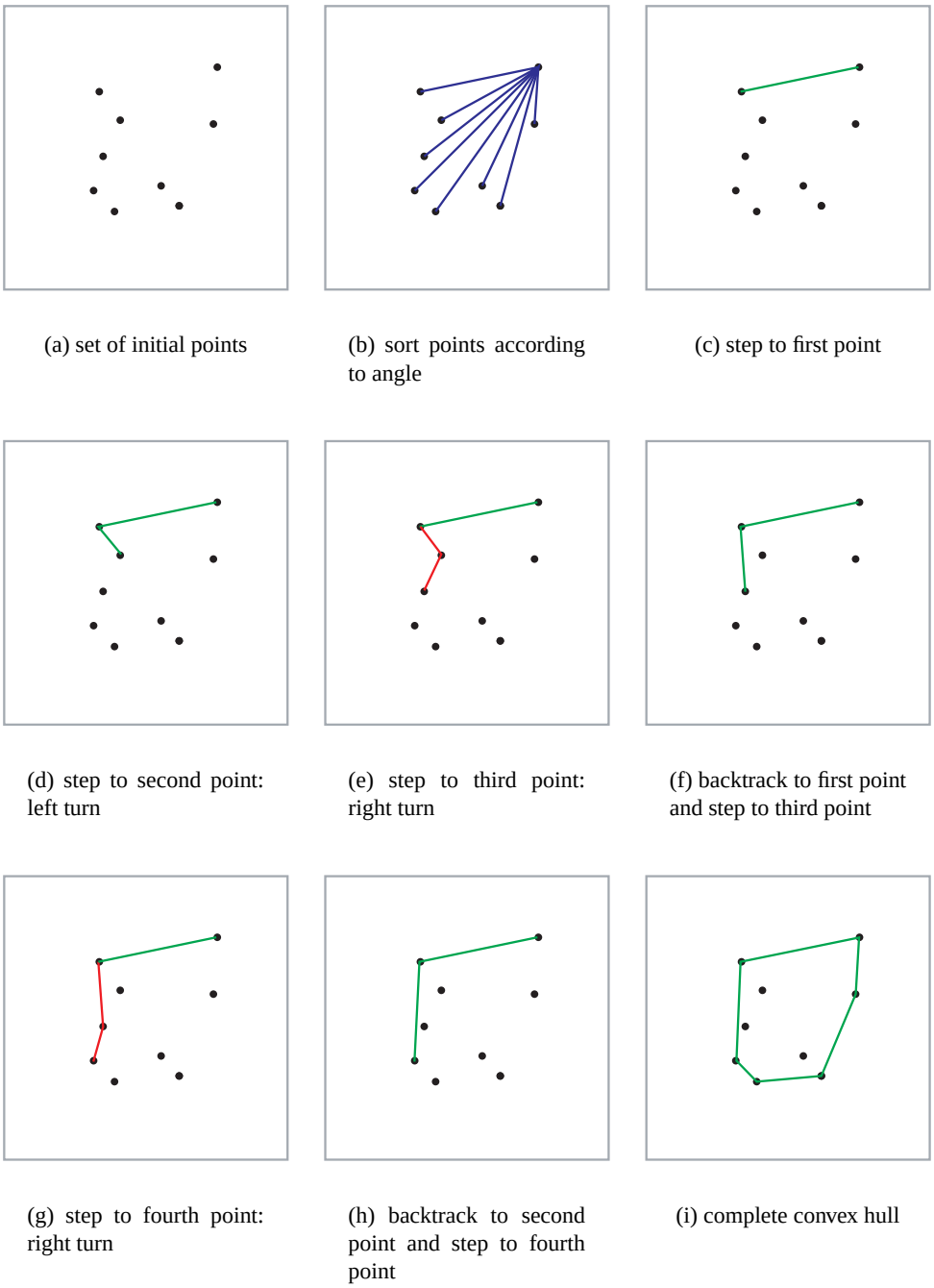
1 An point is chosen that can be used as the pivot. This point should have either a minimal or a maximal x or y-coordinate, so it is guaranteed that it is part of the hull.

2 Sort the points in order of increasing angle around the pivot.

3 Build the hull by traversing the points in this order, adding an edge when we make a left turn, and back-tracking when we make a right turn.

The important steps of this algorithm are depicted in figure II.1.

[1] Adapted from http://www.cs.princeton.edu/ ah/alg_anim/version1/GrahamScan.html.



(a) set of initial points

(b) sort points according to angle

(c) step to first point

(d) step to second point: left turn

(e) step to third point: right turn

(f) backtrack to first point and step to third point

(g) step to fourth point: right turn

(h) backtrack to second point and step to fourth point

(i) complete convex hull

Figure II.1: Graham scan algorithm for calculating the convex hull of a set of points

Appendix III
Technical Details of the Demo Implementation

We will explain some of the technical details of the implementation of our demo for the purpose of future research. Some software has to be installed before you can run the demo:

- either Eclipse Prolog or SWI Prolog,
- the batik SVG-browser (for Linux),
- Internet Explorer with the Adobe SVG plugin (for Windows).

The back-end of our system was programmed using Eclipse, a version of Prolog that was developed by the IC-PARC Imperial College. Later on, some routines were adapted to SWI Prolog, a version of Prolog which was developed at the *Sociaal Wetenschappelijke Informatica* department of the University of Amsterdam (UvA). In the current version of our demo, the main routine includes a predicate called `which_prolog` which checks if it has been loaded in either Eclipse or SWI Prolog and fails otherwise. The software was developed to compile under both Linux and Microsoft Windows, although the win32 implementation of eclipse turned out to be less stable from time to time. The generated SVG files were viewed using the Batik SVG-browser (developed by Apache) under Linux and the Adobe SVG-plugin in Microsoft Internet Explorer for Windows. As of yet, the software runs from the command line. However, currently a web interface is being implemented. The code has been split up into several Prolog files. Here is the list of files the demo consists of:

- `main.pl`: sets the initial state and calls the main loop,
- `focuspointslice.pl`: part of the slicing algorithm that deals with the area around the focuspoint,
- `linearslice.pl`: part of the slicing algorithm that deals with linear slices,
- `fill.pl`: contains the graphics generation predicates,

- `colour.pl`: generates the colour palette,
- `shapes.pl`: contains the shape libraries,
- `draw.pl`: translates Prolog data to SVG,
- `util.pl`: some helpful predicates,
- `compat-swi.pl`: checks for SWI Prolog compatibility,
- `test.pl`: testing presets.

In addition, we have made use of a couple of colour libraries provided by the Cuypers Colorpicker System:

- `hsltorgb.pl`
- `dec2hex.pl`

The output is written to the file `svggen.svg`. In the following sections, we will shortly explain the main implementational issues of the main loop, the slicing algorithm, the colour-generation and shape generation predicates and the output.

1. Main Loop

The `set_global_parameters`-predicate sets the initial state of the Prolog fact database. Besides asserting most of the user-defined variables, such as the depth and the name of the shape set to be used, it also calls a predicate to generate the colour palette and get the list of shape coordinates of the correct set.

When the main parameters have been set correctly, the values for the canvas dimensions, leading line coordinates and recursion depth are passed on to the slicing algorithm, while the other parameters are asserted as global variables. Before the main recursion is started a file-stream is opened to which the SVG data is written. A `header` and `footer` routine have been added that write the standard XML tags at the beginning and the end of the document. The background is initialised with the values of the canvas dimensions and the hexadecimal value of the background colour. The main loop is started by calling the `slice`-predicate:

```
open('svggen.svg',write,FILE),
asserta(svgfile(FILE)),

header,
draw_canvas([CX,CY],BGSCHEME),
slice([0,0,CX,CY],LEADINGLINE,DEPTH),!,
render_leadingline(LEADINGLINE),
footer,

close(FILE).
```

2. The Slicing Algorithm

In principle, the `slice`-predicate takes two simple data structures and a field for de recursion depth as arguments. The data structures represent the rectangular outline of the slice and a the leading line.

The slice outline is represented as a list of four arguments. The first two denote the *x* and *y* value of the top-left corner of the rectangle. This is the anchor point of the slice. The second two arguments contain the width and the height values. Because the first slice is initialised with the state of the entire canvas, the anchor point is equal to the origin while the width and height values are equal to the canvas dimensions.

The leading line representation consists of a list of points. Each point is represented as a coordinate-pair in a sub-list. The convention for the order of the leading line points we have chosen is the following: focus point (`[FPX,FPY]`), left corner point (`[CPLX,CPLY]`) and right corner point (`[CPRX,CPRY]`). When the slice contains no focus point, the first point is discarded. When the slice does not contain a part of the leading line, it is represented as an empty list (`[]`). We will consistently hold on to this notation.

```
slice([X,Y,WIDTH,HEIGHT],[[FPX,FPY],[CPLX,CPLY],[CPRX,CPRY]],N).
```

There are three `slice`-predicates. The first one checks if the maximum iteration value of the loop has been reached. The loops are counted in decreasing order from the user-defined depth to zero. If the loop-termination predicate fails, the algorithm branches out either to a focus point slice or to a linear slice, based on the fact if the length of the leading line has the value 3 or 2 respectively. However, before the slice is subdivided by the `focuspoint_slice` or `straightline_slice`-predicate, the predicates in `fill.pl` are initialised with the current slice values.. This way, before the next recursion-depth is reached, a new layer of graphics is written to the output. When the maximum recursion-depth has been reached, the composition consists of a layered structure of shapes. The graphics generation predicates are discussed in section 3.

The slices that contain the focus point are subdivided into four child slices. The `focuspoint_slice` predicate initially determines in which quarter of the parent slice the point is located, after which the the golden ratio values are calculated. Based on these, two width values and two height values are calculated[1]. Taking all the permutations of the width and height values, we get four different rectangles. If the canvas dimensions comply to the golden ratio, the smallest rectangle will always be a square, while the second smallest will have the same proportions as the largest rectangle.

To simplify the calculation of the new leading line coordinates, we split up the triangular leading line into two linear ones. Our representation guarantees that one of the leading line points is always to the left of the focus point and one is always to the right. Also, because the canvas is divided into four rectangular areas, a vertical guide is always present either to the left or to the right of the focus point. A horizontal guide is always present either above or under the focus point. Added together, these properties guarantee that the leading line is intersected either one or zero times in both dimensions. In two dimensions, the leading line is intersected zero to two times on both sides of the focus point. Crossing the vertical guide twice would only occur when

[1] Remember that, applying the golden rule, A is to B as B is to C; C is equal to the parent width or height. The largest of A and B, say A, is calculated by dividing C by the golden ratio, Φ. B is subsequently calculated by subtracting A from C.

the focus point is either to the left or to the right of both corner points, which is never the case. Because the corner points are initially located on the vertical edges of the parent-slice, the maximum total number of intersections is three: once with the the vertical guide and twice with the horizontal guide.

The `calculate_newLL`-predicate is thus called twice, once for the left and once for the right part of the leading line. This predicate returns a list of intersection points and one of the flags `vert`, `horiz`, `horiz_firts` or `vert_first` to indicate the order. When the lines intersect not once, only the flag `none` is returned. This provides us with sufficient information to deduce which part of the leading line should be assigned to which of the child-slices.

The anchor point coordinates and the slice dimensions are temporarily saved in two lists: `[X,Y,X1,Y1]` and `[W1,W2,H1,H2]`. `X` and `Y` contain the coordinates closest to the origin. `X1` is equal either to the small or to the large golden ratio division of parent slice width. However, this value has been added to the `X`-value of the parent-slice anchor point because of recursion. The same holds for `Y1`. `W1` and `W2` correspond to the width of the left and the width of the right two slices respectively, while `H1` and `H2` correspond to the height of the top and bottom ones. Adding them all together, for each of the four new slices a predicate is called which loops back to `main.pl`. This is done in clockwise order starting from the top-left corner:

```
slice([X,Y,W1,H1],LL,N),     % top-left
slice([X1,Y,W2,H1],LL,N),    % topright
slice([X1,Y1,W2,H2],LL,N),   % bottomright
slice([X,Y1,W1,H2],LL,N).    % bottomleft
```

When one of the child-slices does not contain a part of the leading line, recursion stops, which is the reason that the algorithm sometimes branches out to less than four new `slice`-predicates. The main difference between the `focuspoint_slice2`-predicates, however, is the way the leading line coordinates are re-divided over the new calls to `slice`.

In section 3.1 we have discussed the main ideas behind linear slices. In our implementation, the parent slice is subdivided accordingly. The leading line list consists of two points instead of three:

```
slice([X,Y,WIDTH,HEIGHT],[[CP1X,CP1Y],[CP2X,CP2Y]],DEPTH)
```

The `getminimalslice`-predicate first determines which of the four options of figure 3.5 is optimal. If a new guide cannot be added without intersecting the leading line in either direction, the predicate returns the value `no`. This terminates the recursion loop. Else, `straightline_slice2` calls a `slice` with the values of the child-slice that contains the leading line section.

3. Colour Schemes and Filling Algorithms

Each iteration of the main loop, the `fill_slice`-predicate is called with the values of the current slice node, the current leading line, the slice type and the recursion depth. In addition, a colour is randomly selected from the palette. In `fill_slice`, the global names of the pattern and set are matched and added to the parameter list of the `fill`-predicate, which contains the actual graphics generation steps. In the current implementation of the demo, there are three different fill-types which use single shapes:

single_shape, single_shape_randomalign and single_shape_randomalign_rotate. We will only discuss the last, because it contains the code of the first two.

First, a shape is matched from the shape set randomly with the get_shapes-predicate. It returns the coordinates of the shape and its bounding box. There are two types of shapes: polygons, which consist of actual Prolog lists, and path shapes, which consists of strings containing SVG code. The polygon notation can be used to do calculations on. For example, when a polygon is matched, the bounding box is calculated by the simple get_BB-predicate. In the case of the path shapes, the bounding box has to be set by hand.

```
shape(polygons,[LIST,BB]):-
    LIST = [[0.5,0],[1,0.67],[0,0.67]],
    get_BB(LIST,BB).


shape(paths,[LIST,BB]):-
    LIST =
    ['M50.45,18.602l-16.646,5.113 ... L0.904,49.25z'],
    BB = [100,100].
```

When a shape has been selected, it is re-scaled to fit inside the slice by the rescale_to_fit-predicate. Because the shape minimally fits inside the slice either horizontally or vertically, the direction is returned, which can be important for the re-alignment transformation. The get_rotationvalues-predicate gets a random rotation that is a multiple of 90 degrees and calculates the translation accordingly. Finally, the values are passed on to the draw_transformations-predicate:

```
draw_transformations(
    [TYPE,[COORDS1,BB2]],
    COLOUR,
    [[translation|[TRANSLATION2]],
     [scale,[FACTOR]],
     [translation|[TRANSLATION1]],
     [rotation,[ROTATION]]],
    N).
```

Here, TRANSLATION1 denotes the translation to correct the rotation of the shape and TRANSLATION2 denotes the re-alignment translation. The order in which the different transformations are placed is important. We designed the draw_transformations-predicate to be able to write multiple transformations of multiple types. Therefore, each transformation list has the type as the head and a list of transformation values as tail. With single shapes, the tail will contain only one value and the depth (N) will be set to 1. However, if we want to fill each slice with multiple shapes, we can easily extend this predicate.

The current version of our demo does not contain shape patterns. In chapter 4, we explained how we faced several problems. We solved some of these problems for single shapes by redefining the requirements. For example, we restricted the rotations

to multiples of 90 degrees and discarded skewing. Due to time constraints, we were not able to re-implement the pattern algorithms according to the adjusted requirements. However, an earlier version contained the generation of multiple shapes to some extent. Here, we made use of a list of values for translations and rotations. For the translation values we interpolated between the values of the corner points of the leading line. We determined the rotation step size by dividing 360 degrees by the recursion depth. Re-scaling was done by taking the natural logarithm of the recursion depth[2].

```
shape_pattern(focuspointslice,debug,[X,Y,_,_],COLOUR,
                            [[_,_],[CP1X,CP1Y],[CP2X,CP2Y]],N):-
    get_shapes(SHAPE1),
    ln(N,LN), rescale_points(LN,SHAPE1,SHAPE2),

    H_STEPSIZE is (CP2X - CP1X)/N,
    V_STEPSIZE is (CP2Y - CP1Y)/N,

    R_STEPSIZE is 360/N,

    get_translations(X,Y,H_STEPSIZE,V_STEPSIZE,N,TRANSLATIONS),
    get_rotations(R_STEPSIZE,N,ROTATIONS),

    draw_transformations(SHAPE2,COLOUR,
    [[rotation|ROTATIONS],[translation|TRANSLATIONS]],N).
```

For the linear slice pattern, we took a rotation step size of 180 degrees and switched the order of the rotations and translations. In addition, we made use of small skewing values.

4. Output: Drawing Functions

The `draw.pl` file includes all the predicates that are concerned with the translation of prolog code to SVG-syntax. At startup, in the `main`-predicate a new file is created:

```
open('svggen.svg',write,File),
```

where `svggen.svg` is the name of the output file. The corresponding Prolog instance is matched by the variable `File`. We assert the fact `svgfile(File)`, so we are able to open the file stream from another predicate by matching `svgfile(SVGfile)` and then calling a `write`-predicate with `SVGfile` as argument. As was mentioned earlier, before the main loop is entered a predicate is called that writes the standards XML header lines. These include the version, the SVG DTD and the SVG namespace. Before termination, a footer-predicate closes the file with the tag <\svg>.

The predicates for rendering the proportional guides and leading line are dependent on the command line input parameters and are strictly meant for visualising the underlying structure of the framework. The input parameters are asserted in the main predicate. The rendering predicates in `draw.pl` simply match these global values. The `draw_proportions`-predicate is called at the moment the location of the new guides is know. In the focus point-slicing algorithm, this is right after the orientation of the

[2]The bounding box was not used yet.

focus point has been established and the measurements of the child-sliced have been calculated accordingly. For linear slices, this is done in the predicate `straightline_slice2`. The `renderProportions` predicate decides if the guides should rendered and if so, if all lines are rendered the same width or with gradual increments. Also, the colour of the lines should be specified.

The most important predicates are those that write the SVG syntax for shapes and transformations. First of all, a single point is only written in the case the intersection points with the leading line are rendered. The `draw_pointlist`-predicate, which traverses a list while writing the contained point coordinates in SVG-syntax, is called when we generate the coordinates of a shape. It is used by `draw_polygon`. This predicate draws an entire shape, including fill and stroke. Because we generally do not apply strokes to our shapes, we have implemented an extra predicate, `draw_shape`, which sets the `stroke` argument to `'none'` and the `stroke-width` to zero.

The `draw_transformations`-predicate is used to write the shape patterns. It recursively draws a number of shapes, dependent on the depth variable `N`, and encapsulates it in a transformation group. Because it should be possible to do multiple transformations at once, a second predicate `draw_transformations2` is needed to parse multiple lists of transformation-coordinates in parallel. For each of the transformation types (rotation, translation, scale and skew) a list of values should be specified. At the head of the list, the type should be indicated as a term. A list for a shape that is subsequently rotated and skewed would look like this:

```
[[rotation|ROTATIONS],[skewX|SKEWSX],[skewY|SKEWSY]]
```

where the names in capitals denote the lists of transformation values. Note that the skew transformation takes two arguments and therefore needs lists of values for both *x* and *y* coordinates. A resulting SVG shape nested inside a transformation group would look like this:

```
<g transform="rotate(264.0) translate(733.3,242.0) ">
<polyline points="-54.1,-108.3 54.1,-108.3 108.3,-54.1 108.3,54.1
                  54.1,108.3 -54.1,108.3 -108.3,54.1 -108.3,-54.1 "
fill="#6db428" stroke="none" stroke-width="0"></polyline>
</g>
```

The values for `draw_transformations` are calculated by the shape pattern algorithms. Finally, some predicates are included to write the polygons shapes and paths.

Appendix IV
Acronyms

AI Adobe Illustrator

AI Artificial Intelligence

CAD Computer Aided Design

CMYK Cyan Magenta Yellow blacK

CWI *Centrum voor Wiskunde en Informatica*

DOM Document Object Model

DTD Data Type Definition

EPS Encapsulated PostScript

GIF Graphics Interchange Format

GIS Geographic Information System

HTML HyperText Markup Language

HSL Hue Saturation Lightness

JEPD Jointly Exhaustive and Pairwise Disjoint

JPEG Joint Photographic Experts Group

PDF Portable Document Format

PNG Portable Network Graphics

RCC Region Connection Calculus

RGB Red Green Blue

SVG Scalable Vector Graphics

SWI *Sociaal Wetenschappelijke Informatica*

SMIL Synchronized Multimedia Integration Language

XHTML eXtensible HyperText Markup Language

XML eXtensible Markup Language

XSL(T) eXtensible Stylesheet Language (Transformations)

References

1. Alexander, C., *Notes on the Synthesis of Form*, Harvard University Press, 1964.
2. Alexander, C., Silverstein, M., Ishikawa, S. *A Pattern Language*, Oxford University Press, 1977.
3. Alexander, C., *The Timeless Way of Building*, Oxford University Press, 1979.
4. Allen, J.F., "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, 26(11):509-521, 1983.
5. Angel, E., *Interactive Computer Graphics, a Top-down Approach with OpenGL*, second edition, Addison Wesley Longman, Inc., 2000.
6. Albertazzi, L, "The Aesthetics of Particulars: A Case of Intuitive Mechanics", *Axiomathes*, Nos. 1-2, pp169-196, 1998.
7. Badros, J. et al., "A Constraint Extension to Scalable Vector Graphics", *Proceedings of 10th International World Wide Web Conference*, pp489-49, May 2001.
8. Bal, H. E., Grune, D., *Programming Language Essentials*, Addison-Wesley, 1994.
9. Behrens, R.R., "Art, Design and Gestalt Theory", *Leonardo: Journal of the International Society of Arts, Sciences, and Technology*, Vol. 31, Number 4, pp299-304, August/September 1998.
10. Booth, D.J., *Foundation Discrete Mathematics for Computing*, International Thomson Computer Press, ISBN 1-85032-276-7, 1995.
11. Cha, M.Y., Gero, J.S., "Shape Pattern Representation for Design Computation", in progress.
12. Chang, D., Dooley, L., Tuovinen, J.E., "Gestalt Theory in Visual Screen Design - A New Look at an Old Subject", *WCCE2001 Australian Topics: Selected Papers from the Seventh World Conference on Computers in Education*, Vol. 8, McDougall, A., Murnane, J., Chambers, D. (eds), Conferences in Research and Practice in Information Technology, Copenhagen, Denmark, ACS, pp5-12, 2002.
13. Chase, S.C., "Representing Design With Logic Formulations of Spatial Rela-

tions", *Workshop Notes, Visual Representation, Reasoning and Interaction in Design*, Fourth International Conference on Artificial Intelligence in Design, Stanford University, June 1996.

14. Cheadle, A.M., Harvey, W., Sadler, A.J., Schimpf, J., Shen, K., Wallace, M.G., *Eclipse, an Introduction*, Technical Report IC-PARC-03-1, IC-PARC Centre for Planning and Resource Control, William Penney Laboratory, Imperial College London, 2003.
15. Coates, P.S., Appels, T., Simon, C., Derix, C., *Current work at CECA*, Generative Art Conference 2001, Centre for Environment & Computing in Architecture, 2001.
16. Cohen, H., "Parallel to Perception: Some Notes on the Problem of Machine-Generated Art", *Computer Studies*, IV-3/4, 1973.
17. Cohn, A.G., Randell, D.A., Cui, Z., "Taxonomies of Logically Defined Qualitative Spatial Relations", *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Guarino, N., Poli, R. (eds), Kluwer, 1994.
18. Cohn, A.G., Hazarika, S.M., "Qualitative Spatial Representation and Reasoning: An Overview", *Fundamenta Informaticae*, 46, 1-2, pp1-29, 2001.
19. Cohn, A.G., Gotts, N.M., "The 'Egg-Yolk' Representation of Regions With Indeterminate Boundaries", *Proceedings, GISDATA Specialist Meeting on Geographical Objects with Undetermined Boundaries*, Burrough, P., Frank, A.M. (eds), pp171-187, Francis Taylor, 1996.
20. Hofstadter, D.R., Dennett, D.C., *The Mind's I - Fantasies and Reflections on Self and Soul*, ISBN 0-14-006253-X Basic Books, Inc. 1981.
21. Devetakovic, M., *Communicating Generic Process - Some issues of Representation Related to Architectural Design*, Generative Art Conference 2001.
22. Dorin, A., "Physicality and Notation, Fundamental Aspects of Generative Processes in the Electronic Arts", *Proceedings of First Iteration*, Dorin & McCormack (eds), CEMA, Melbourne, pp80-91, December 1999.
23. Drakengren, T., Jonsson, P., "Eight Maximal Tractable Subclasses of Allen's Algebra with Metric Time", *Journal of Artificial Intelligence Research*, 7, pp25-45, 1997.
24. Gips, J., "Computer Implementation of Shape Grammars", *Workshop on Shape Computation*, MIT, 1999.
25. Honour, H., Fleming, J., *Algemene Kunstgeschiedenis*, 9th, revised & extended edition, Meulenhoff Amsterdam, 2000.
26. Johnsson, P., Drakengren, T., "A Complete Classification of Tractability in RCC-5", *Journal of Artificial Intelligence Research*, 6, pp211-221, 1997.
27. King, M., "Computers and Modern Art: Digital Art Museum", *Proceedings of the Fourth Conference on Creativity & Cognition*, Loughborough, UK, pp88-98, October 2002.
28. Kirsch, J.L., Kirsch, R.A., "Computer Grammars for the Syntactical Analysis of Paintings", *World Art - Themes of Unity in Diversity*, Acts of the XXVIth Interna-

tional Congress of the History of Art, Lavin, I. (ed), 1989.

29. Krokhin, A., Jeavons, P., Jonsson, P., *Reasoning about Temporal Constraints: Classifying the Complexity in Allen's Algebra by using an Alebraic Technique*, Programming Research Group, Oxford University Computing Laboratory, PRG-RR-01-02, 2001.
30. Luger, G.F., Stubblefield, W.A., *Artificial Intelligence - Structures and Strategies for Complex Problem Solving*, Second Edition, The Benjamin/Cummings Publishing Company, Inc, ISBN 0-8053-4785-2, 1993.
31. Meyer, J.C., Van Der Hoek, W., *Epistemic Logic for AI and Computer Science*, Van Rijsbergen, C.J. (ed), Cambridge University Press, ISBN 0-521-46014-X, November 24, 1995.
32. Nack, F., Manniesing, A., Hardman, L., "Colour Picking - the Pecking Order of Form and Function", *Proceedings of the eleventh ACM International Conference on Multimedia*, ACM Press, Berkeley, USA, November 2 - November 8, 2003.
33. Ossenbruggen, J.v., *Processing Structured Hypermedia - A Matter of Style*, Phd Thesis, Vrije Universiteit, Amsterdam, The Netherlands, April 10, 2001.
34. Rome, E., "Simulating Perceptual Clustering by Gestalt Principles", *8th International Symposium on Intelligent Robotic Systems - Proc. of the 25th Workshop of the Austrian Association for Pattern Recognition ((AGM/AAPR) (AGM 2001))*, Scherer, S. (ed), Oesterreichische Computer Gesellschaft, Vienna, ISBN 3-85403-147-5, pp191-198, 2001.
35. Van Der Vrie, E.M., et al., *Combinatoriek, Grafen- en Getaltheorie*, special edition, Open Universiteit, Heerlen, ISBN 90-358-1509-2, 1996.
36. Visser, A. de, *Hardop Kijken: een Inleiding tot de Kunstbeschouwing*, Ch.3, pp61-80, SUN, Nijmegen, ISBN 90-6168-2517, 1986.
37. *Scalable Vector Graphics (SVG) 1.0 Specification*, Ferraiolo (ed), W3C Recommendation 2001.
38. Wertheimer, M., "Untersuchungen zur Lehre von der Gestalt - II", *Psychologische Forschung 4*, pp301-350, 1923.
39. Whitehead, A.N., *Dialogues*, Greenwood Publishing Group, Reprint edition, ISBN 08-371-9341-9, February 25, 1977.
40. Williams, R., *The Non-Designer's Design Book*, Peachpit Press, 1994.