İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İŞLEMSEL TASARIM: TASARIM ARACI OLARAK MATEMATİK

YÜKSEK LİSANS TEZİ

Büşra GÜLER

Bilişim Anabilim Dalı

Mimari Tasarımda Bilişim Programı

ARALIK 2017

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İŞLEMSEL TASARIM: TASARIM ARACI OLARAK MATEMATİK

YÜKSEK LİSANS TEZİ

Büşra GÜLER
(523151005)

Bilişim Anabilim Dalı

Mimari Tasarımda Bilişim Programı

Tez Danışmanı: Prof. Dr. M. Birgül ÇOLAKOĞLU

ARALIK 2017

İTÜ, Fen Bilimleri Enstitüsü'nün 523151005 numaralı Yüksek Lisans Öğrencisi Büşra GÜLER, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "İŞLEMSEL TASARIM: TASARIM ARACI OLARAK MATEMATİK" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. M. Birgül ÇOLAKOĞLU** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Leman Figen GÜL** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Arzu ERDEM Kadir Has Üniversitesi	.....

Teslim Tarihi : 17 Kasım 2017
Savunma Tarihi : 18 Aralık 2017

ÖNSÖZ

Evren yaşayan bir organizmadır; nefes alır, etkilenir, tepki verir, dönüşür.. Bu yaşayan organizmanın içinde etrafımızda görebildiğimiz, cansız olarak tanımladığımız cisimler dahil olmak üzere herşey değişim ve dönüşümleri ile yine canlılık semptomu gösterirler. Mimarlar olarak dokunduğumuz şehirler, mekanlar, yüzeyler işte bu cansız dediğimiz, fakat çevreleriyle bitmeyen bir etkileşim içinde olan canlılardır. Bu canlıların çevreleri ile olan etkileşimlerinin ana kaynağı, çevreleri ile olan "*ilişki"*leridir. "*ilişki*"nin başladığı yerde "*düzen*" başlar. "*düzen*" ise kendini ancak "*matematik*" ile gösterir. Öyleyse sabit, ilişkisiz, düzensiz, hissiz şehirler, mekanlar, yüzeyler; "*canlılar*", oluşturmak doğayla çatışma içindedir. Bu mütevazi çalışmanın yegane arzusu, bu fikri, eline çalışmayı alan kişiye aktarabilmektir. Fikirler bir günlük, bir yıllık bir vaktin sonucu değil, bir yaşamın, bir aklın yansımasıdır ve burada şükranlarımı hayatın her köşesinde cebime bir şeyler bırakan herkese sunmak isterim.

Kitapları ile kütüphanemi dolduran Aldous Huxley, Cemil Meriç, Douglas Adams, Gavin Francis, George Orwell, Jean-Christophe Grange, Richard Dawkins, Rıfat Ilgaz, Sait Faik Abasıyanık, Umut Sarıkaya, Yevgeni Zamyatin ve diğer isimler bilmeseler de fikirlerime katkıları büyüktür. Süreçte benim yoğun programıma, anlatım dilime, inatçı fikirlerime alışan ve çalışmama katkıları olan tez hocam Prof. Dr. Birgül Çolakoğlu'na teşekkür ederim.

Tüm akşam sohbetlerimiz için ve yapmak istediklerim için beni cesaretlendiren Melike Altınışık'a desteklerinden dolayı teşekkür ederim.

Mesleki fikirlerim, büyük katkıları olan hocalarım Gudjon Thor Erlendsson, Prof. Dr. Gülen Çağdaş, Gregor Veiss sayesinde oluşmuştur. Kitapları ile Prof. Dr. Robert Aish, Mark Burry, Casey Reas, Paul Coates, Ben Fry işlemsel tasarımı sevmemi sağlayan isimlerdir.

Ve "teşekkür ederim" demenin minnetimi yeterince ifade edemeyeceği kişiler.. Manen her daim yanımda olan dostlarıma müteşekkirim. Herşeyin yegane sebebi, mutluluklarım, heyecanlarım, hayattaki tüm duygularım, annem ve babama minnet borçluyum, iyi ki varsınız.

Kasım 2017

Büşra Güler
(Mimar)

İÇİNDEKİLER

Sayfa

SEMBOLLER

cos(u) **:** Cosinüs

d **:** Yoğunluk

f **:** İki kütle arası çekim kuvveti büyüklüğü

g **:** Evrensel çekim sabiti

m **:** Kütle

r **:** Yarıçap

sin(u) **:** Sinüs

q **:** State (durum)

π **:** Pi sayısı (3,14)

Δ **:** Fark

ŞEKİL LİSTESİ

Sayfa

İŞLEMSEL TASARIM: TASARIM ARACI OLARAK MATEMATİK

ÖZET

"Tanrı fikirlerini hesapladığında dünya yaratıldı" Gottfried Leibniz (Rocker, 2006, s.18'de atıfta bulunulduğu gibi).

"Mimarlar fikirlerini hesapladığında, herşey algoritmaya dönüştü" (Rocker, 2006, s.18).

İnsanın temel gayesi ve uğraşısı evreni anlamak ve çözümlemek üzerinedir. Tüm bilim dalları evrenin yasalarını araştırarak, doğadan öğrendiği ile üreten insana klavuzluk etmektedir. Mimari tasarımın ifade ediliş biçimi olan formun klavuzluğunu geometri, matematik, fizik bilimleri yapmaktadır. Bu sebepledir ki form arayışında geometri, antik çağlardan beri mimarinin temel araştırma konusu olmuştur. Phaedrus, geometrinin evrenin biçimini anlamak için, insan zihninin bir ürünü olduğunu söyler (Corbusier, 1948). Bu açıdan düşünüldüğünde geometri için "formun dilidir" ifadesini kullanmak yanlış olmayacaktır. Geometrinin matematikle ilişkisi göz önünde bulundurulduğunda ise "matematiğin geometrinin dili" olduğu söylenebilir. Dolayısı ile dilini form ile ifade eden mimarinin matematik ile kesin bağı bu yolla oluşmuş olur.

Günümüzde matematik; işlemsel tasarım düşünce sistemi ile bir dil olmanın da ötesine geçip "tasarımın aracı" konumuna gelmiştir. Böylelikle matematiksel dille türetilen form sisteminin sağladığı esneklik; nihai form değil, formu üretme bilgisini içinde barındıran süreç tasarımının oluşmasına olanak sağlamıştır. Sonuç olarak ortaya çıkan ürün kadar, ürünün oluşma süreci, sürecin tasarımı, sonuç ürününün üretilebilirliği bilgisi, mimaride yeni pratikler olarak karşımıza çıkmaya başlamıştır. Başka bir ifade ile tasarım tasarlanmaya başlamıştır. Yeni alanlar, matematiğin rolünün tasarım sürecinde başkalaşması ile oluşmuştur. Matematiğin başrolde olduğu tasarım fiilinde, yeni bir düşünce sistemi ve yeni bir dil oluşmaya başlamıştır. Bu düşünce ve dil sistemine matematiğin mutlaklığının tesiri, tasarımın kurallara tabi olmasını sağlamıştır. Tasarım sürecinin kurallar dizisi ile oluşturulması algoritmik düşünceyi adres göstermektedir (Jabi, 2013, s.22). Algoritmik düşünceyi, kurallar zinciri ile formu oluşturan bilginin deşifre edilmesi olarak tanımlayabiliriz. Dijital çağda algoritma ile form üretebilme olanağı, formu üretme tanımlarını değiştirmiş, form "*yapılmayan*", kurallara veya algoritmalara bağlı olarak "*bulunan*" olmuştur (Agkathidis, 2015, s.17). Bu bağlamda, tasarım fiilinde yeniler ve yenilenenler, başka bir deyişle tasarım sürecinde algoritmik düşünce sisteminin oluşturduğu potansiyeller, keşfedilmesi gereken bir konu haline gelmiştir.

İşlemsel tasarımın özüne inilmek istendiğinde matematik ile kaçınılmaz bir iletişim süreci başlar. Bu kapsamda tezde form ile tasarımcı arasında, matematiksel dil aracılığı ile kurulan iletişim süreci mercek altına alınmıştır. Geometrinin dili olan matematik, tasarım aracı olarak keşfedilmeye çalışılmıştır. Matematiksel dil ile aranan, tasarlanan form, matematik ile anlaşılmaya çalışılıp, formu oluşturan adımlara, tasarım sürecinde iken müdahaleler nasıl yapılır anlaşılmaya çalışılmıştır. İşlemsel tasarımın aracı olarak

görülen matematik ile formun kontrolü nasıl yapılabilir, işlemsel tasarım yöntemleri ile form üretiminde oluşabilecek kısıtlamalar ve imkanlar nelerdir gibi sorulara yanıt aranmıştır.

Sonuç olarak tezde temel olarak hedeflenen, işlemsel tasarım ile değişen tasarım dilinin form üretme üzerindeki etkilerini araştırmaktır. Kurallı bir dil olan matematik ile tasarlama, tasarım süreci ve düşüncesini doğrudan değiştirmiştir. Tez kapsamında öncelikli olarak yeni tasarlama eyleminin doğurduğu düşünce sistemi irdelenmiştir. Akabinde bu yeni düşünce sisteminin dili tartışma konusu olmuştur. Tüm bu araştırmalar, aslen işlemsel tasarımın, başlı başına bir düşünce sistemi olarak görülmesi sebebi ile yalnızca mimari mecrada değil, pek çok farklı disiplinin penceresinden yapılmıştır.

Tartışılan konuların reel dünyada yansımaları ise mimari alan çerçevesi içinde ele alınmıştır. Tez kapsamında bu bağlamda, bir kabuk tasarımı önergesi ile işlemsel tasarım süreci örneklemesi yapılmıştır. Tasarıda ideal tek bir form üretiminden ziyade form üretmek için algoritmalarla sistem oluşturulmuş olup form üretimi formülize edilip, işlemsel tasarım yaklaşımı olarak düşünülen, "form aranandır" savı somutlaştırılmaya çalışılmıştır. İşlemsel tasarım düşünce sistemini oluşturan "akıl" ve "dil" temellerine dayandırılarak, bu sistemin gereği dil olan kodlama ile üretilen modelin, matematik ile kurallı geometri oluşturmayı, tasarımın yapılabilme bilgisini, süreç tasarımını anlamada etkili olduğu düşünülmektedir.

COMPUTATIONAL DESIGN: MATH AS A DESIGN TOOL

SUMMARY

"When God calculates and exercises his thought, the world is created" Gottfiried Leibniz (Rocker, 2006, s.18 as in case).

"Today, when architects calculate and exercise their thoughts, everything turns into algorithms!" (Rocker, 2006, s.18).

The main purpose and the interest of humanity is understanding and analysing the universe. While studying the laws of the universe, entire disciplines are guiding people that produce by learning from the nature. The guidance of the form which is expression of architectural design, is made by geometry, mathematics and physics. For this reason, geometry in search of form, has been the fundamental research field of architecture from the ancient age. Phaedrus says that geometry is the creation of human mind, to perceive the world (Corbusier, 1948). With this approach, the statement of the "language of form"should not be a mistake. While the relations of geometry and mathematics is considered, it could be said that "mathematics is the language of geometry". With this way, the relationship between mathematics and architecture that describes its language through form, is formed.

At the present time, mathematics becomes a "design tool" not a design language, with the computational design idea. The reason of that, the technological evoluation and digital revolution. With digital design tools, we started to communicate directly with datas and this communication can be only with algorithmic thinking. Algorithmic thinking is based on mathematical thinking, so all new terminologies and relations are connecting architecture and mathematics. As a result, not just in architectural field, in all areas, humanity starts to handle the technology with the power of accuracy of mathematics.

In architecture, mathematical systems are started to create to solve design problems. In other words, design approaches turn out from abstract ideas to controllable datas. With the helping of working with datas, design process started to be designed. Information types are changed and new types are occured. Thus, flexibility that is provided by form system derived from mathematical language, enables designing a process that contains the information of producing form. So, as well as the significance of final product itself, with computational design, also process design starts to be discussed.

The process of the emergence of product, the design of process, the information of the producibility of final product become new fields in the practice of architecture. For this reason, as architects we are facing with these new fields. In other words, design starts to be designed. These new fields are composed by the evolution of the role of mathematics in design process. In addition to that, a new system of thinking and a new language start to be occured in design process which is dominated by mathematics. The influence of the absoluteity of mathematics to these system of thinking and system

of language ensure that the design is subject to the rules. The creation of the design process with set of rules array is addressing of algorithmic thinking (Jabi, 2013, s.22). Algorithmic thinking can be defined as decoding information which creates form with set of rules. In digital age, the possibility of generating form with algorithms, changes the design process. Form is no longer being "made", but "found", based on set of rules or algorithms (Agkathidis, 2015, s.17). In this context, news and reneweds, in other words, the potentials of the algorithmic thinking in the design process, have become a field that must be discovered.

When it is desired to enter into the spirit of computational design, it is necessary to create a connection with mathematics, because inevitable communication process begins with mathematics. For this reason, the communication established between the form and the designer through mathematical language has been under consideration. Generating form with mathematical language is the starting point of this thesis. Mathematics that is the language of geometry is tried to be discovered as a language of design. Searched form by mathematical language, designed form is tried to be understand with mathematics. How the steps that generate form can be manipulated during design process is tried to be discovered. How form can be controled with mathematics that is seen as computational design tool? What is opportunities and constraints of form generation with method of computational design? Responses of such these questions have been searched under computatioal design field.

As a result, the essence purpose of this thesis is searching influences of design language which is changing by computational design on form generation. Design by mathematics which is a formal language is changing directly to the design process and design idea. In the content of thesis, as a first step, the thinking system which is generated by new design action was examed. Then the language of this new system of thought has been the topic of discussion. All of these researches were originally made from the window of many different disciplines, not only in architectural area, because of the fact that computational design is seen as a system of thought in itself. That's why as a system of thought, computational thinking is subject of all disciplines, creates new practises in all disciplines and become a major field to search and study by experts. Computational design is just the response of computational thinking in the field of architecture.

After deep understanding of thinking system and the comminication way of this thinkig, reflections of controversial issues in the real world of computational thinking are handled within the frame of architectural space. Within the context of the thesis, a shell design proposal and a transactional design process sampling were done with computational design techniques.

A form generator system was designed during the shell design process. The system is set uped with algorithms to generate form rather than creating one ideal form. The approach of "form is searched" is embodied. Because rather than modeling a just one form, to reach the optimized form, an algorithmic system was created using Rhino Grasshopper plug-in. This system is founded by the computational thinking and formal languge which are seen as the bases of computational design. With this manner, the algorithm which is used for the shell design can be seen as a system to show ruled geometry generation by mathematics, the information of the feasibility of design and process design which are discussed in the thesis in an effective way.

As a result, humanity is in a new age and living in a data driven spaces. New age created new fields and new business models. The disciplines that is the part of the new

world can be survived. With this manner, what is the way of adaptation to the new system should be the path which all disciplines should follow. In architecture, computational design offers the way to connect architects to this path. For this reason, in fact computational design is a platform in today's architecture, not just an expert field. To communicate with this platform, the thinking behind the approach must be scrutinised. In that time, mathematics shows itself as a foundation of computational thinking and computational designers face with mathematical thought and language. This thesis is placing itself as a guide to first deep meeting with computational design.

1. GİRİŞ

“Form bulmanın önkoşulu formsuz olmaktır ve forma sahip olma koşulunu askıya almak olacaktır” (Ballantyne, 2010, s.8).

0 ^{o}C: su katıdır, şekli katılaşmanın başladığı an oluşmaya başlar

20 ^{o}C : su sıvıdır, şeklindeki değişim temas ettiği yüzey(ler)/zamandır

100 ^{o}C: su gazdır, şekli insan gözünün ayırt edemeyeceği haldedir. Şeklindeki değişim yine temas ettiği yüzey(ler)/zamandır.

Suyun şeklini belirleyen ortam şartları değiştirilip, su sonsuz forma dönüştürülebilir, sonsuz sayıda form tarifi yapılabilir. Fakat değişmeyen şey suyun içeriğidir (yapısı-tanımı). Şekli ne olursa olsun su iki hidrojen bir oksijen atomundan meydana gelir ve bütün fiziksel hallerinde tanımı yine aynıdır.

Bütün bilim dalları inceleme yaptığı mecrada oluşumları çözümler ve bu çözümleri üretmede yöntemler geliştirir. Mimaride işlemsel tasarım, aslen temel olarak inceleme alanımız formu çözümleme niyetinde olan bir düşünce sistemi ve formun nasıl oluştuğunu anlamamızı sağlayan yöntemler bütünüdür. Form soyut bir dil ile matematiksel olarak tarif edilir. Tüm nihai değerler formun tanımından çıkarılıp, oluşma adımları tek tek işlemlere dökülür; bir çizginin uzunluğu 5 cm değil n dir ve sonsuz bir olasılıklar kümesi bu anlatımın içindedir. Formun nihai değerlerinin n ile tanımlanmasıyla, bir kural tanımı yapılmaktadır. Bu da işlemsel tasarımın temelindeki algoritmik düşünceyi gösterir. Sonuç olarak matematiğin üretme alt yapısı aracılığı ile işlemsel tasarım, form arayışının önemli ve belirgin altlığını oluşturma potansiyelleri ile dinamik bir alan olmaktadır.

Bir tasarım düşüncesi olan işlemsel tasarım, ortaya çıktığı yıllardan itibaren, form ve tasarım üretme süreci yeniden tanımlanmaya başlanmıştır. Yeniden yapılan tanımlamaların içerisinde tasarım için yeni olan bazı kavramlar matematik için hiç de yeni değildir. Yeni olan, tasarım sürecinin matemetiksel dil ile tanımlanmaya başlanmasıdır. Kartezyen düzlemde koordinatları *(x1,y1,z1)* ve *(x2,y2,z2)* değerleri olan iki nokta ve bu iki noktanın arasında pek çok noktanın yan yana gelmesi sonunda

((xn,yn,zn), (xm,ym,zm)) çizgi tezahür eder. Bu tanımda formun nerede, ne kadar uzunlukta olduğu anlatılmamıştır, ancak ve ancak bir çizgi tarifi yapılmış, form açık bir netlikle oluşturulmuştur. Tasarımın gizliliği kodlamanın esneksizliği ile ihlal edilmiştir (Burry, 2011). Tasarımcının black box olarak tanımladığımız tasarlama süreci deşifrelenmeye başlanmıştır. Bu da beraberinde Casey Reas in sözünü ettiği üzere, olası tasarım alanını keşfetmek için form üretiminde bir belirsizlik sistemi kurulsa da sistemin kontrol edilebilirliği ile aslında spesifik tasarım probleminin en iyileştirilmiş net sonucuna ulaşmayı sağlamaktadır.

İşlemsel tasarım tek bir sonuç ürününün modellenmesi yerine kendi kendini modelleyebilecek sürecin tanımının yapılmasıdır (Scheurer ve Stehling, 2011). Bu tanımın yapılması formun özüne; geometrisine dönülmesine bağlıdır. Geometrinin adımlar halinde matematiksel bir sistem ile oluşturulması gerekmektedir. Bu da işlemsel tasarım ile tasarımcı - matematik arasındaki mesafenin eridiğini göstermektedir. Tasarımcı artık matematiği bir araç olarak kullanarak, formunu aradığı, deneyimlediği bir süreç içerisindedir. Süreçte bilinmezlikler tasarımcının fikir ve eylemlerinden, matematiğin hesaplanmayı bekleyen değerlerine bürünmüştür.

Bir düşünce sistemi olarak okunduğunda, işlemsel düşüncenin mimarlık disiplini içerisinde yer bulduğu alan işlemsel tasarım olmuştur. Dolayısı ile işlemsel tasarım düşünce sistemini yalnızca tasarım alanı çerçevesinde okumak, kapsam ve sınırlarını yanlış tanımlamaya sebep olacaktır. Bu sebeple işlemsel düşünce sisteminin felsefi ve teorik alt yapısı farklı disiplinler aracılığı ile irdelenmelidir. Kavramların reel karşılıklarını ise mimari alan içerisinde aramak, işlemsel düşüncenin işlemsel tasarım alanında sınırlarını oluşturacaktır. İşlemsel tasarımın temelini oluşturan işlemsel düşünce Jeannette M. Wing in de desteklediği gibi ne bir dil, ne bir araç, ne de bir akımdır, başlı başına bir düşünme sistemidir ve her disiplinden insanın hayatında kökleşecektir.

1.1 Tezin Amacı

İşlemsel düşüncenin tasarımdaki yansıması bugün tartıştığımız bir konudur. Ancak işlemsel düşüncenin gerekliliği olan kavramlar anlaşılıp, bu kavramlar bambaşka konular olarak değil, tasarımın bir parçası olarak tartışılmaya başlandığında, işlemsel tasarım içselleştirilip, doğru bir şekilde kullanılmaya başlanacaktır (Peters, 2013). Çalışmanın amacı öncelikli olarak işlemsel düşünceyi anlamak daha sonra bu

düşüncenin tasarımdaki karşılığını kavramlar üzerinden araştırmaktır. İşlemsel tasarım alanı ile tasarımın dil bulduğu hal olan tasarım süreci bu tartışmada yerini alan önemli bir konu olacaktır.

Tezde işlemsel - hesaplamalı tasarım düşünce sistemi ile yeniden tanımlanan "tasarlama fiili" araştırma konusudur. Formun anlam ve yapımının değişimi işlemsel tasarımın temeli olarak görülmektedir ve işlemsel tasarım ile tasarım sürecinin değiştiği, matematiksel bir alt yapı ile form oluşturulduğu savunulmaktadır. Bu bağlamda, algoritma ile formüle edilen tasarım sürecinde algoritmik düşüncenin ne olduğu ve kodlama dili anlaşılmaya çalışılacaktır. Tasarım aracı olarak düşünülmüş matematik ve kodlama ilişkisi form bulma sürecinde incelenecektir. Teorik olarak araştırılmış kavramlar üzerinden kodlama ile form üretim modeli geliştirilecektir. Form üretiminde, kodlama ile matematik dilinin nasıl forma dönüşüp tasarım aracı haline geldiği gözlemlenecektir. Üretilen modellerin değerlendirmesi yapılmadan form üretme sistemi kurmak hedef olacaktır. Model sisteminin oluşmasında matematiğin birinci dereceden model üzerindeki etkilerini gözlemlemek önem taşıyacaktır.

1.2 Tezin Kapsam ve Sınırları

Tezin ilk bölümünde işlemsel tasarım düşünce sistemi araştırılmıştır. Bu düşünce sistemi sayesinde bir araç haline gelmiş matematik ile form üretimi hedeflenmiştir. Tezin başlangıcı hesaplama, işlemsel düşünme, algoritma gibi kavramların araştırmasıyla yapılacaktır. Yapılan okumalar doğrultusunda bu kavramlar tasarım süreci içerisinde değerlendirilecektir. Tezin ikinci bölümünde işlemsel düşünme, soyut düşünme ve bu düşüncenin dili; işlemsel - betik dil mantığı anlatılacaktır. Bu bölümden yapılan çıkarımlar sonucu matematiksel form tanımı ve geometrik karşılıkları işlemsel form türetme modeli üzerinden değerlendirilecektir. Nihai bir form yerine form arayışı içerisinde olunan sistem içinde değerlendirilmesi yapılacak olan, ortaya çıkan sonuçlar değil, sonucu üretme süreci olacaktır.

1.3 Yöntemi

Tez kapsamında işlemsel düşünce üzerine farklı disiplinler aracılığı ile okumalar yapılarak farklı disiplinlerin yaklaşım ve ürünleri incelenecektir. İşlemsel düşüncenin

tasarımdaki karşılığı literatür araştırmaları yapılarak kavramlar üzerinden tartışılacaktır. Bu düşüncenin tasarımdaki yansıma aracı olan betik dili ve matematik ile ilişkisi örnekler üzerinden değerlendirilecektir. Dijital ortamda geliştirilecek form türetme modeli ile matematik formülleri üzerinden form arayışları yapılarak işlemsel tasarım kavramları somutlaştırılacaktır. Sonuç ürünleri tez kapsamında temel araştırma konusu olmadığı için ürünleri üretme sürecinin değerlendirilmesi yapılmalıdır.

2. AKIL

“Benim odaklandığım makinenin kendisi değil, aklıdır” (Papert, 1980, s.9).

Kaldıraçlar, basit makinelere verilebilecek ilk örneklerden biridir. Basitçe yük ile kuvvetin pozisyonlarına göre nasıl değiştiğini deneyimleyebileceğimiz sistemlerdir. Makas, kapı, tahterevalli, tartı, örneğini uzatabileceğimiz birçok alet kaldıraç prensibi ile; kuvveti ve yükü arasındaki ilişkiye göre çalışır. Eğer bu tek prensip örneklerini sayfalarca yazabileceğimiz birçok alet ve makinenin ortak paydası ise, paydayı bilmek paya gelebilecek tüm değişkenleri bilmekle eş değerdir. M.Ö. 3. yüzyılda yaşamış Arşimed’in bulduğu *kuvvet x kuvvet kolu = yük x yük kolu* prensibi ile sonsuz kaldıraçlar kümesi oluşmaktadır (Url-1). Bu prensibin tanımında ne makas ne kapı denilmiştir. Makas ve kapı gibi pek çok aletin çalışma mantığı bir satırda özetlenmiş, aynı mantık ile çalışabilecek, keşfedilmemiş, isimsiz pek çok aletin de tanımlaması yapılmıştır. “Bana sağlam bir destek noktası verin, dünyayı yerinden oynatayım” sözü ile Arşimed de tek bir prensip ile bu sonsuz olasılıklar alemini tasavvur etmektedir (Url-2). Dünyanın sağlam destek noktasına uzaklığı ile ilişkili olarak onu kaldırmak için gerekli kuvveti bulmak, düzeneğin tanımını yapan formülde değerleri yerine koymaktır. Bu sayede sonsuz olasılıkta sistem düşünmek mümkün olmuştur.

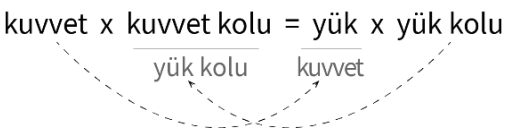
2016 yılının kasım ayında Londra’da gerçekleşen ASCAAD Parametricism & Materialism konferansında oturum başkanı Wassim Jabi konuşmasında işlemsel tasarımı akla dönüş olarak anlattı. “Saatin nasıl göründüğü ile değil nasıl çalıştığı ile ilgileniyoruz” diyen Jabi, işlemsel tasarım ile tasarımın aklını sorgulama ve irdeleme sürecinde olduğumuzu dile getirmektedir. Peki tasarımın aklı nedir, neden akla dönülmektedir ve varılmak istenen yer neresidir?

Evrende bulunan formlar ancak oluşma prensipleri veya oluşma mantıkları ile anlaşılabilir (Tsigkari ve diğ, 2011). Bilimsel yasalar formüllerdir ve gerçek dünyanın genel özelliklerini anlamamızı sağlarlar. Eğer anlaşılmak istenen spesifik bir gerçeklik ise yapılması gereken spesifik durumun değerlerini bu bilimsel formüllerde yerine yazmaktır. Dünya ile ay arasındaki çekim kuvveti bilinmek isteniyorsa yapılması gereken, Newton’un evrensel çekim yasası formülünde, $f = m1 \times m2 \times g/d2,$ ay ve

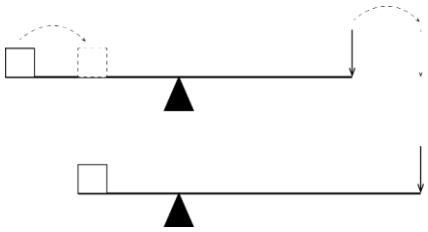
dünya için gerekli değerleri yerine yazmaktır (Borgmann, 2011). Bir tasarımın prensibi veya mantığı, tasarımın oluşum sürecidir. ASCAAD konferasında Robert Aish "Herhangi birşey tasarlayabilirim ama eğer işlemsel tasarım yapmak istiyorsam nasıl tasarladığımı düşünmek zorundayım; tasarım sürecimi düşünmek zorundayım" ifadelerini kullanarak tasarım için akıl, süreçtir demiştir.

Akıl ile ilgilenme sebebimiz açıktır; tasarımın ifadesi olan formu anlamak. Scripting Culture kitabında "Kuralları yıkmadan önce bilmeniz gerekir" diyen Mark Burry de aklı anlamayı işaret etmektedir. Tasarımın aklı; süreci anlamak, süreci adım adım sistematize etmekle mümkün olabilir. Geleneksel tasarım yöntemleri ile tasarım problemine göreceli tasarımcı yanıtları, blackbox dediğimiz tasarımcı içgüdüsü, işlemsel tasarımda kağıda dökülen somut verilere dönüştürülerek bilinmeyenler ortadan kaldırılmaya çalışılmaktadır. Bu saydam süreç kurulumu, tasarımda kontrolü destekleyen bir model oluşturmayı sağlar. Robert Aish'in anlatımı ile "Sonucun büsbütün kontrolünde olmak isteyen tasarımcı, sürecin kontrolünde olmalıdır" (Burry, 2011, s.67'de atıfta bulunulduğu gibi). Tasarlamak, sistem kurgulamaktır ve fizik, matematik formülleri ile tanımlanan sistemlere de tasarım gözü ile bakmak, başka bir ifade ile tasarıma işlemsel formüller ile tanımlanabilecek süreç olarak bakmak, işlemsel tasarımın temelini oluşturmaktadır.

Sonuç olarak işlemsel düşünce yaklaşımı ile çalışma kuralını bildiğim makinem için en işlevli makineyi nasıl yapabileceğimi tartışabilirim; en az kuvvetle en çok yükü nasıl taşıyabileceğimi. *kuvvet x kuvvet kolu = yük x yük kolu* formülünün içinde *yük / kuvvet = kuvvet kolu / yük kolu* formülü bulunmaktadır ki bu da kuvvetten elde ettiğim kazancın oranını verir (Şekil 2.1). Daha az efor sarf ederek kaldırmak istediğim bir yük düzeneği için yapmam gereken bellidir; destek noktasına olabildiğince uzak bir yerden kuvvet uygulamak, yükü de bu noktaya olabildiğince yaklaştırmak (Şekil 2.2). Makinemin şekli bu noktalar belirlendikten, gerçekten en iyi şekilde çalışan bir sistem kurulduktan sonra oluşacaktır. Sonuçta ancak aklın çözümlenmesi ile anlamanın ve kontrol etmenin reellinden söz edilebilecektir.



Şekil 2. 1: Kuvvet Formülü.



Şekil 2. 2: Kuvvet formülü diyagramı.

2.1 İşlemsel Düşünce

Akla dönmek bir başlangıç adımıdır. Aslen akıl nasıl bir düşünce sistemi ile anlaşılır sorusuna yanıt aramak gerekmektedir. Araştırmacı filozof Aaron Sloman "İşlemsel düşünce tüm zekaların doğasını ele alır, mikrop zekası, hayvan zekası, insan zekası, makine zekası" demiştir (Sloman, 2012, s.7). Jeannette Wing de "İşlemsel düşünce makine zekası ile yüz yüze gelmektir: İnsanlar makinelerden daha iyi ne yapabilir, makineler insanlardan daha iyi ne yapabilir? Asıl soru ise işlemsel - hesaplanabilir nedir?" sözleri ile karşımıza "işlemsel düşünce" kavramını çıkarmaktadır (Wing, 2006, s.33). 1980 yılında ilk kez Seymour Papert işlemsel düşünce kavramını ortaya atmıştır (Papert, 1980). 244 sayfalık Mindstorms kitabında sadece bir kez işlemsel düşünce diyen Papert, bundan ancak 16 yıl sonra 1996 yılında An Exploration in the Space of Mathematics Educations (Matematik Eğitimi Alanında Bir Araştırma) makalesinde işlemsel düşüncenin açıklamasını yapmıştır. Ancak işlemsel düşünce ne 37 yıllık geçmişi olabilecek bir kavram ne de bugün ilk kez tartıştığımız bir fikirdir. Alman matematikçi Gottfried Leibniz'in "Tanrı fikirlerini hesapladığında dünya yaratıldı" sözü üzerine, işlemsel düşünce evrenin oluştuğu saniye, bundan yaklaşık 14 milyon yıl önce, evrene ayak basmıştır demek yanlış bir ifade olmayacaktır (Url-3).

Aklın tabiatı, onun yapı ve işleyişi konuları 1920'lerde psikolog Jean Piaget tarafından ele alınmaya başlanmıştır. Piaget'in yaptığı zeka tanımlarından biri zekanın, yaşayan ve eylemde bulunan bir "işlemler sistemi" olmasıdır (Günçe, 1971). Piaget çocukta zeka yapılanmasını 4 evreye ayırır. İşlemler sistemi de Piaget Teorisine göre 3. ve 4. evrelerde gelişmektedir. Piaget, 3. evreyi somut işlemler dönemi, 4. evreyi soyut işlemler dönemi olarak adlandırmaktadır ve çocukta 6 yaş itibari ile geliştiğini söylemektedir (Url-4). Somut işlemler döneminde mantıksal düşünme, sayıları kullanma, problem çözme, muhakeme yapma gibi yetiler gelişirken soyut işlemler döneminde soyut düşünme, çok yönlü algı, analiz-sentez yapma gibi yetenekler kazanılmaktadır (Url-5). Piaget teorisine göre işlemsel öğrenme süreçlerinin var

olduğunu görürüz. İşlemsel öğrenme varsa ozaman işlemsel düşünce insanın doğasında vardır sonucuna varabiliriz.

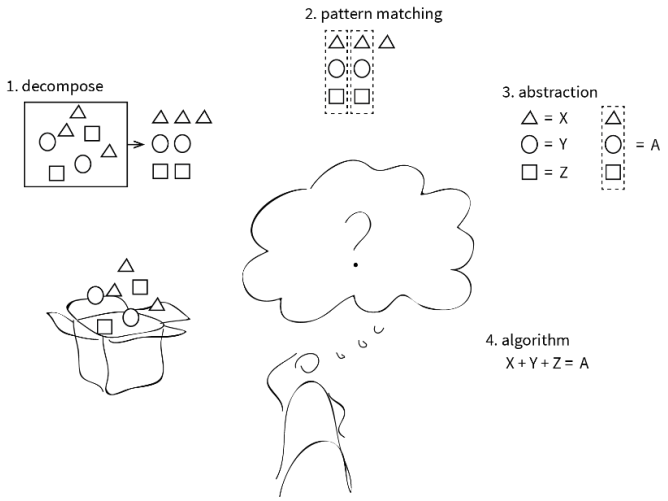
Epistemoloji, bilgi felsefesi, öncülerinden Seymour Papert, Marvin Minsky, neredeyse keşfedilir edilmez bilgi üzerine düşünmede yeni bir yöntem olabileceğini farkettikleri bilgisayar ve bilgi felsefesi üzerine çalışmalar yapmaya başlamış ve bilgisayarın sadece çok güçlü bir hesap makinesi, veri işleyicisi olmasının yanında öğrenme adına çok önemli bir araç olduğunu savunmuşlardır (Coates, 2010). Matematikçi, eğitimci, bilgisayar uzmanı Papert'ın bilgisayar ve bilgi üzerine başlayan çalışmaları işlemsel düşünce kavramının ortaya çıkmasındaki ilk adımlar olmuştur. Papert Mindstorms kitabına "Bilgisayarlar insanların düşünme ve öğrenmesini nasıl etkiler" sorusu ile başlamakta ve bilgisayarı bir "öğrenme makinesi" olarak görmektedir (Papert,1980, s.5). Çocukların bilgisayarla iletişim kurabilmek için bilgisayarı programlamayı öğrenmesi gerektiğini savunur. Ancak temel sorusu "Bilgisayar mı çocuğu kodlar yoksa çocuk mu bilgisayarı" olmuştur (Url-6). Bu sorusu üzerine yaptığı araştırmalar sonucunda "Bilgisayar ile iletişim kurmayı öğrenmek, diğer öğrenmeleri de değiştirecektir" iddiasını ileri sürer (Papert, 1980, s.25). Bu da aslında bilgisayarın öğrenme fiilini etkilediğini, Papert'ın kendi ifadesi ile, "bilgisayarın insanı kodladığı.." sonucuna vardığını gösterir. Ancak bu ifadeleri insan doğasında mevcut olan işlemsel düşüncenin bilgisayar ile hayatımıza girdiği değil, bilgisayarın insandaki işlemsel düşünceyi desteklediği yönünde okumak gerekmektedir. Papert'ın 1996 yılında matematik eğitiminin bilgisayarın eğitimde rol alması ile geliştirilebileceğini savunması matematiksel dil ile bilgisayar dili olan programlama dilinin birbirine yakınlığını göstermektedir (Papert, 1996). Bir başka ifade ile insandaki işlemsel zeka ve bilgisayarlar ile iletişim dili olan programlama zekası paralellik gösterir diyebiliriz. Sonuç olarak bahsettiğimiz "akıl", "işlemsel düşünce", "matematiksel zeka", "programlama" gibi tüm kavramların birbirleri ile iç içe ve birbirleri ile beslenen kavramlar olduğu sonucuna varabilir, sadece tasarım değil tüm disiplinlerde sistemi anlamak; sistemin "aklını" "işlemsel düşünce" ile anlama ve "matematiksel" bir dil ile çözümleme sürecidir diyebiliriz.

İşlemsel düşünce büyük bir çoğunluk için betik yazmaktır. Bir düşünce yapısını sadece bir eylem ile tanımlamak yüzeysel bir tanımlamaya sebep olacaktır. Alan Kaye, Seyomur Papert ve Marvin Minsky gibi isimlerin başlattığı işlemsel düşünce kavramı, 2006 yılında bilgisayar mühendisi Jeannette Wing ile popüler bir konu olmaya

başlamıştır. Wing'e göre, "İşlemsel düşünce, bilgisayar biliminin temel kavramları ile problem çözmek, sistem tasarlamak ve insan davranışlarını anlamaktır" (Wing, 2006, s.34). Wing işlemsel düşüncenin temel özelliklerini şu şekilde sıralamıştır:

- *Programlanan değil, kavramsallaştırılandır;* Çok katmanlı soyut düşünmeyi gerektirir.
- *Temeldir, bir beceri değil;* Tüm insanlığın bilmesi gereken bir temeldir, bir makinenin rutini değildir.
- *Bilgisayarın değil insanın düşünme şekli;* İnsanların problem çözme yöntemidir, bilgisayar gibi düşünmeye çalışmak değildir.
- *Tamamlayıcı, matematiksel ve mühendislik zekayı birleştirici;* Matematiksel düşünce gerektirir ve temelleri tüm bilimlerde olduğu gibi matematiğe dayanır. Mühendislik düşüncesini gerektirir, gerçek hayat ile iletişim kuran sistemler inşa edilir.
- *Eser değil fikir;* İşlemsel düşüce yaklaşımı ile çözülen problemler hayatı düzenler, hayatın içinde insanlar ile iletişim ve etkileşim halindedir
- *Herkes ve heryer için;* İnsanın tüm çaba ve girişimlerini bütünleyici bir gerçekliktir.

Wing kısaca işlemsel düşünceyi herkesin "bilgisayar mühendisi gibi düşünmesi" olarak yorumlamıştır. Bahsettiği özellikler bilgisayar mühendisliği disiplini için temeller olsa da sadece bu disiplin ile ilişkilendirmek tekrar sınırlı bir tanım yapılmaya başlanması demek olacaktır. Wing de 2008 yılında "İşlemsel düşünce analitik düşüncedir. Problem çözme yaklaşımları matematiksel düşünce ile ilişki kurar. Gerçek dünya sınrları içinde kompleks sistemler tasarlama yaklaşımları mühendislik düşüncesi ile ilişki kurar. Hesaplanabilirlik, zeka, akıl ve insan davranışlarını anlama yaklaşımları bilimsel düşünce ile ilişki kurar" yorumunu yapmıştır (Wing, 2008, s.3717). Bu yeni tanım ile Wing işlemsel düşüncenin sınırlarını daha da genişletmiş ve geliştirmiştir diyebiliriz. İşlemsel düşünce, bilimden insanlığa neredeyse tüm disiplinlerden etkilenmekte ve tüm disiplinleri de etkilemektedir (Bundy, 2007). Yapılan tanımlar işlemsel düşünce için yeni kavram ve adımları ortaya çıkarmıştır; analiz (decompose), örüntü tanıma (pattern matching), soyutlama (abstraction), algoritma (algorithm) (Url-7) (Şekil 2.3).



Şekil 2. 3: İşlemsel düşünce adımları (yazar tarafından yorumlanmıştır).

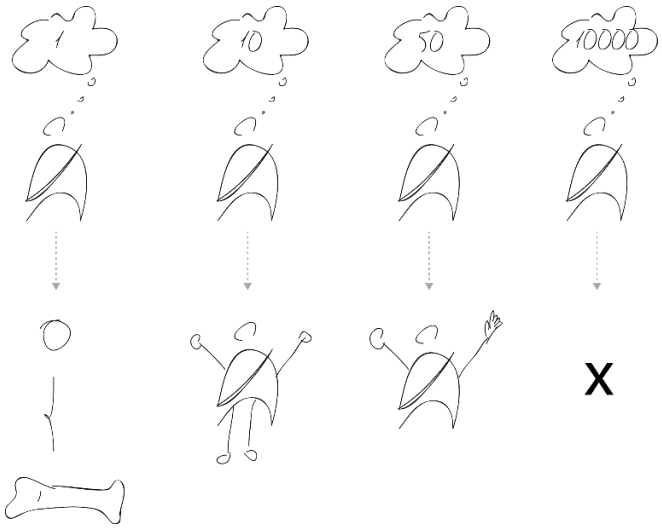
İşlemsel düşünce ile kompleks bir problem çözümünde öncelikli olarak büyük problem "analiz" adımında küçük alt problemlere bölünür ve çözümlenebilir alt başlıklar çıkarılır. "Örüntü tanıma" adımında yanlızca kompleks problem üzerine odaklanmak yerine tüm bu alt problemler incelenip benzer problemler göz önüne alınarak çözümler üretilir ve bu sayede benzer problemlerin her birine bir seferde cevap aranmış olur. Spesifik durum, özellik ve bilgilerin elenerek çok sayıda probleme tek cevap üretilmesi, çok sayıda durumun ortak paydasını bularak bu payda altında toplamak "soyutlama" adımında gerçekleştirilir. Algoritma adımında alt problemlere üretilmiş çözümlerin sistematize edilerek adım adım kompleks problemin çözümü için kurallar dizisi oluşturulur (Url-8-9). Bu adımlar bir makine, bir robot için tanımlanmamıştır aksine tamamen insanı merkez alır; "Bilgisayar bir makine olabilir, ama bir insan da olabilir. Diğer bir deyişle işlemsel düşünce makinaya ihtiyaç duymaz" (Wing, 2008, s.3719).

2.1.1 Soyut düşünce

"Akıl, soyut düşünceyi sürdürme becerisidir" Lewis Madison Terman (Pfeifer ve Scheier, 2001, s.6'da atıfta bulunulduğu gibi).

Hesap "calcul" dediğimiz zaman, sözcüğün kendisi bizi uzak çağlardan gelen bir yönteme gönderir. Latince "calculus" sözcüğü "küçük çakıl" anlamına gelir. Sayıların icadından önce ilkçağ insanı sayı kullanma ihtiyacı içerisindedir ve soyutlama zekası ile yazı yazmayı bilemezken sayıları yazar olmuştur (Ifrah, 1995, s.13). Matematik öğretmeni Georges Ifrah öğrencilerinin kendisine sorduğu "Efendim, rakamlar nereden geliyor?" sorusu üzerine yaptığı 20 yıllık araştırmalar sonucunda, Rakamların

Evrensel Tarihi kitabı ile rakamların ve hesabın çakıl taşlarından bilgisayar çağına tarihini anlatmaktadır. Soyut düşüncenin ilk örnekleri olarak yorumlanabilecek rakamlar öncesi dönemde sayma bir eşleme sistemi üzerine kurulmuştur. İlkçağ insanı sayılar yerine mağara duvarlarına, ağaç dallarına çentikler atmış, ipleri düğümlemiş veya çakıl taşlarını kullanmıştır. Bir nesneyi temsil eden çizgiler, çakıl taşları, insanın el ve ayak parmakları zaman içerisinde saymak istedikleri nesnelerin çokluğu karşısında yetersiz kalmış bu kez semboller ve işaretler devreye girmiştir. Bilinen en eski sayma sistemine sahip eski Mısırlılar, rakamlar yerine sembol ve harfler kullanmışlardır. Romalılar Romen rakamları olarak bildiğimiz sistemi bir elin parmaklarından ilham alarak üretmişlerdir. Bugünkü modern medeniyetin meydana gelmesini sağlayan "0" ın ilk önce Hintliler tarafından bir sembol olarak, daha sonra Arap sistemi ile matematik işlemlerindeki bugünkü icadı sayesinde, ancak günümüzde kullandığımız modern rakam sistemi büyük ölçüde oluşturulmuştur (Ifrah, 1995). Sonuç olarak az miktarda nesneyi başka objelerle eşleyerek, işaretleyerek, daha sonra sayılması gereken miktar arttıkça vücudundaki uzuvları kullanarak sayabilen insan, sayılması gereken miktara karşılık gelebilecek nesneler, çizgiler, parmaklar yetersiz kaldığında semboller keşfetmeye başlamış ve bu süreç rakamların icadı ile son bulmuştur (Şekil 2.4). Tüm bunlar gösteriyor ki, işlemin dili olan rakamların icadı, soyut bir mantığın temelleri üzerine gelişmiş bir süreç olmuştur. Buradan hareketle tüm düşünme sistemlerini işlemsel düşüncenin paydası altında toplayan, işlemsel düşünmenin şartı, "soyut düşünme"dir demek yanlış bir ifade olmayacaktır.



Şekil 2. 4: Rakamların Evrensel Tarihi kitabındaki anlatım ve görsellere istinaden yazar tarafından yorumlanmıştır.

Alfabeler soyutlamanın, ifadeleri sembolize etmenin gerçek bir karşılığıdır. Mors alfabesi yazıları, çizgi ve noktaların farklı kombinasyonları ile şifreleyerek ses, ışık, radyo dalgası olarak gönderilebilir hale getirmede kullanılan bir haberleşme dilidir. 1836 yılında elektronik telgrafı icat eden Samuel Finley Breese Morse'un ilk planı yanlızca sayıları kullanmak olsa da Alfred Vail tarafından harflerin de eklenmesi ile mors, İngilizceden sonra en çok kullanılan dil oluşmuştur (Url-10). Verici cihaz ile mors alfabesine göre çizgi ve noktalar ile yeniden oluşturulan metin, telgraf koluna çizgi için uzun, nokta için kısa basışlar yapılarak alıcı cihaza gönderilir. Haflerin, nokta ve çizgiye soyutlanması belirli bir kurala göre yapılmıştır (Şekil 2.5). Böylece nokta ve çizgiler ile harfler, kelimeler ve cümleler oluşur.

Şekil 2. 5: Morse Alfabesi, harflerin oluşum şeması (http://cryptomuseum.com/radio/morse/, Temmuz 2016).

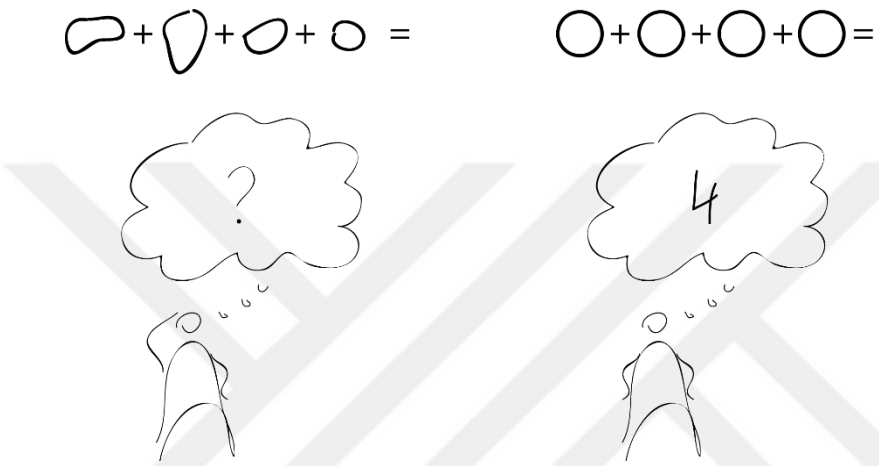
Soyut düşünme felsefe alanında da karşımıza çıkmaktadır. Isaac Newton 1687'de yazdığı Principia Mathematica kitabında felsefede akıl yürütmeyi kurallarla anlatmıştır ve soyut düşüncenin bir dizi anlatımını yapmıştır diyebiliriz. Dört kural altında topladığı akıl yürütmenin kurallarından birini şöyle açıklamıştır: "Aynı doğal tesirleri izah etmek için, mümkün olduğunca, aynı sebepleri bunlarla ilişkilendirmeliyiz. Bir insanda ve bir hayvanda solunum; taşların Avrupa'da ve Amerika'da alçalma hareketi; ocağımızdaki ateşinin ışığı ile Güneş'in ışığı; ışığın Dünya'da yansıması ile gezegenlerde yansıması gibi..." (Newton, 1687, s.21). Bu

kural ile görülen, işlemsel düşüncenin bir adımı olarak kabul ettiğimiz "soyut düşünce"dir. Sebep, sonuç, mekan, zaman gibi faktörler elenerek, bir nevi olaylar kümesinin ortak elemanı bulunur, birbirleri ile benzer olaylar bu ortak elemanın başlığı altında bir sınıflandırma oluşturur. 1921 yılında Educational Psychology (Vol. 12, s, 123-147, 195-216) dergisinde aklın tanımı 14 psikoloğa sorulmuş ve aralarında Lewis Madison Terman da soruyu "Soyut düşünceyi sürdürme becerisidir" olarak yanıtlamıştır (Pfeifer ve Scheier, 2001, s.6'de atıfta bulunulduğu gibi). Bu tanımlamaya göre de aklın temeli olarak görülmüş "soyut düşünce"; düşünmenin, anlamanın, irdelemenin, çözümlemenin temelidir diyebiliriz.

Soyutlama "abstraction" sözcük kökeni latince abs-trahere birşeyden çekme, uzaklaştırma (to draw away) anlamına gelip, insan algısında meydana gelen bir aktivite olması itibari ile temeli insan olan tüm disiplinlerin sözlüğünde yer alan bir kavramdır (Saitta ve Zucker, 2013). Başka bir ifade ile, bir şeyin detaylarını çıkarmak ve genellemektir diyebiliriz. Burada birşeyin bilgisinde eksiltme yapma bilgide kayıp anlamına gelmemelidir. Doğru bir soyutlama kritik bir noktadır; çok detaylı bir anlatım karışık ve anlaşılamaz bir anlatıma sebep olurken, fazla soyutlama, amaç için gerekli bilginin anlatımına engel olur (Kramer, 2007). Jeannette M. Wing' göre soyutlama, işlemsel düşünce ile hangi detayların vurgulanıp hangi detayların göz ardı edileceğini tanımlama sürecidir (Wing, 2008). Soyutlama o şeyin esasını anlamada basitleştirmek, özü kendi üst katmanlarından süzmektir (Goldstone ve Barsalou, 1998) . Jeff Kramer "Hesaplamada soyutlama anahtar mıdır?" adlı makalesinde soyutlama, gereksiz ayrıntıları kaldırma ve değişkenlik içinde ortak bir "öz" tanımlama becerisidir der (Kramer, 2007). Tüm bu yorumlara göre, rakamlarda, alfabelerde, fizik ve kimya kanunlarında, felsefede, psikolojide, temeli insana dayanması sebebi ile tüm bilimlerde soyutlama, özü oluşturan temel ilişkiler ağını deşifre etme, başka bir ifade ile aklı anlama düşüncesi, "akıl"a dönüştür. Soyutlamalar, işlemin zihinsel aracı, aklıdır (Wing, 2008).

Saymayı öğrenmek insanın işlemsel düşüncesinin başlangıcıdır, ve doğal olarak cebir ile işlemsel düşüncenin ardından aritmetik, hesaplama ve soyutlama, sembol temelli düşünce, oluşmuştur. Sayma, aritmetik, semboller ve soyut düşünme hesaplamanın temelini oluşturur (Henderson ve diğ, 2007). Jeannette M. Wing soyutlama işlemsel düşüncenin temelidir der ve işlemsel düşünce tanımlamalarında her zaman soyutlama kavramını da tanımlamalarına dahil ederek hesaplamada, kavramları zaman ve

mekanın ötesinde soyutladığımızı söyler (Wing, 2008). 17. yüzyılın önemli düşünürlerinden John Locke, soyutlama olarak yorumlayabileceğimiz "genel fikri bulma"yı zihnin bir icadı olarak anlatır ve kendisi için bu bir süreçtir. Zihinde ilk önce oluşan şeyler algı için sıradan ve tanıdık olan belirgin fikirlerdir. Böylelikle belirli fikirler önce alınır ve seçilir. Bununla birlikte spesifik olan fikirler daha sonra, özel fikirler de en son zihin tarafından algılanır. Bu aslen detaylarından arındırarak temel fikrin tanımlamasını yaptığımız soyutlamanın, algılama sürecinde ilk etapta oluştuğu şeklinde yorumlanabilir (Şekil 2.6).



Şekil 2. 6: Soyutlama (yazar tarafında yorumlanmıştır).

John Locke soyut düşünce fikrini esasen anlamamızı sağlayacak bir üçgen tanımı yapmıştır; "Üçgen ne dik ne eğik, ne eşkenar ne çeşitkenar olmalıdır; bunların hepsidir ve aynı zamanda hiçbiri" (Locke, 1690). Üçgen denildiğinde, dik, eşkenar, ikizkenar, çeşit kenar, tüm üçgen çeşitlerini kapsamış oluruz. Ancak bir üçgen aynı anda hem dik üçgen hem eşkenar üçgen olamayacağı için de, üçgen dediğimiz şey, tüm üçgenleri kapsarken bir yandan elimizdeki üçgenin hangi üçgen olduğu bilgisi indirgenmiş olur. Bu kez tüm bu farklı tip üçgenleri tek tek ele almayarak, aynı doğru üzerinde olmayan üç adet noktasının olduğu ve bu noktaları birleştiren doğrulardan meydana geldiği ortak bilgisi, sadece üçgen tanımı ile bir kelime altında toplanmış olur. Soyutlama tam da Locke'nin üçgen tanımında olduğu gibi bir aradalıktır. Çoklu katmanlar ile çalışma, bu katmanlar arasındaki ilişkiyi anlamaktır. Jeannette Wing soyutlama sürecindeki bu katmanları soyutlama fonksiyonu, simülasyon ilişkisi ve dönüşüm, "eşleme süreci" olarak anlatır (Wing, 2008). Lorenza Saitta ve Jean-Daniel Zucker bu katmanları daha basit ifade ile "genelleme" (generalization), "yaklaştırma" (approximation) ve "yeniden düzenleme" (reformulation) olarak tanımlarlar (Saitta ve Zucker, 2013). Tüm

bu adımlar aslında birbiri ile iç içe olup genel amaç "basitleştirme" dir. Soyutlama katmanlarını nörolog David Marr'ın Vision kitabında kompleks bilgi ve sistemlerin anlaşılması konusunda verdiği bir örnek üzerinden okumak mümkün;

> ..bir şişede bir miktar gazı düşünelim. Sıcaklık, basınç, yoğunluk ve bu faktörler arasındaki ilişkilerin yani termodinamik etkilerin tanımının tek bir denklem ile formüle edilmesi mümkün değildir. Bu tür etkiler kendi seviyelerindeki parçacık ilişkileri ile anlatılır. Yani mikroskopik, makroskopik seviyeler vardır. Eğer bir kişi bir şişe gazı veyahut sinir sistemini, bir bilgisayar programını, herhangi kompleks bir sistemin tam olarak anlaşılmasını umuyorsa, ozaman o kişi farklı seviyelerde düşünmeye hazır olmalıdır. (Marr, 1982, s.20).

2.2 Tasarımda İşlemsel Düşünce

Vitruvius, Roma Cumhuriyeti dönemi sonlarına doğru, M.Ö. 90-20 yılları arasında yaşadığı tahmin edilen, ilk Roma imparatoru Augustus'a ithafen, kuvvetle M.Ö. 25 sıralarında yazdığı De Architectura ile bilinen mimar-mühendistir. Mimarlık Üzerine On Kitap olarak bugün derlenmiş bu en eski klavuz niteliğindeki kitapta Vitruvius, bu kadar eski bir dönemin mimarlığını anlatırken bugün için yöntemler, sınıflandırmalar, kavramlar eskidir ancak düşünce yapısı canlılığını korur niteliktedir. Bu sebepledir ki tasarımda işlemsel düşünceyi Vitruvius'un sözlerinde bulmak mümkündür. Herşeyden önce Vitruvius mimarlığın dil kökeninin Yunanca "düzen", "düzenleme" kelimesinden geldiğini dile getirir. Mimarlık Vitruvius için "kural düzenlemesine" bağlıdır (Vitruvius, M.Ö. 1. Yüzyıl, s.9). Uyum (fitness) ve güzelliğin (beauty) de düzenin beraberinde geldiğini şu sözleri ile aktarır; "Uyum ve güzellik kişilerin yargısındadır. Ne zaman ki yükseklik genişlikle, genişlik uzunlukla uyuşursa, başka bir deyişle, herbiri birbiri ile uyuşursa, kişiler uyum ve güzelliği bulabilir" (Vitruvius, M.Ö. 1. Yüzyıl, s.10). Vitruvius'dan etkilenerek 1443'te yazdığı De Re Aedificatoria, 10 Kitapta Leon Battista Alberti'nin Mimarlığı, kitabında İtalyan ressam, şair, filozof, müzisyen, mimar Alberti ise uyum ve güzelliği ahenk (harmony) olarak şu şekilde tanımlamaktadır;

> ..bir binanın tüm parçaları ahenk içinde olmalıdır. Binanın her bir parçası diğer tüm parçalar ile uyum içinde bütünün güzellik ve başarısına katkı sağlamalıdır. Bir bina diğer parçalar ihmal edilirken sadece bir parçası ile güzel olamaz; bütün, tek ve iyi eklemleri olan bir vücut gibi görünebilmek için birbirleri ile ilişki içinde olmalıdır. (Alberti, 1443, s.63).

Buradan yola çıkılarak mimarlığın, tasarımın ilk kez anlamının arandığı, bilebildiğimiz kadarı ile en eski yıllarda yapılan tanımlamalar içinde işlemsel fikri

yakalamak mümkündür. “düzen”, “uyum”, “ahenk”, “parça-bütün” gibi kavramlar işlemsel tasarımda matematiğin mutlaklığı ile “düzenlenen” lerdir. Jeannette Wing’in işlemsel düşünce tanımına tekrar dönülürse, işlemsel düşünce çoklu katmanlarla çalışma, bu katmanlar arasındaki ilişkiyi anlamaktır (Wing, 2008). Uyum ve ahenk, birlikte çalışan tasarım elemanlarının ilişkisini anlamak ve düzenlemek ile bütünde okunabilir. Igor Stravinsky’nin bir sözüne alıntı yapmak gerekirse “Ağaçlarda rüzgarın uğultusu, nehrin dalgaları, kuşların sesi müzik değildir, ancak “müziğin vaadidir”. Tonlar ancak düzenlenmesi meziyeti ile müzik olabilir” (Url-11). İlişkileri tanımlanmamış ve birbirleri ile çalışmayan parçalar, 20. yüzyılın önemli bestecilerinden Stravinsky’nin ifadesindeki sesler olarak kalır, müzik oluşamaz.

William Mitchell, Stravinsky’nin “..Tonların ancak düzenlenmesi meziyeti ile müzik olabilir” sözünü “sanatçının malzemesine form vermesi” olarak yorumlamıştır ve bunu Platon ilkesi ile bağdaştırmıştır (Mitchell, 1990). Platon fiziksel objelerin, kusursuzun, soyut düşüncenin, kusurlu taklitleri olduğunu düşünür. Mitchell’in yorumundaki malzeme, tonları, form verme eylemi, düzenlemeyi işaret ederken müzik, tasarımın kendisidir. Temelde söylenmek istenen, tasarımın parçalardan ve adımlardan oluştuğu, bu adımların kurallar sisteminde düzenlenmesi ile nihai ürüne ulaşıldığı, sürecin işlemsel düşünce ile geliştiğidir. Bu fikri Platon’un ilkesi ile bağdaştırması, tasarımda soyut düşünceye işaret etmesi olarak düşünülebilir. Başka bir ifade ile, işlemsel düşüncenin temelleri ile oluşan tasarım eyleminde, aslen işlemsel düşüncenin özü olması sebebi ile “soyut düşünce” mevcuttur yorumu yapılabilir. Mitchell, tasarımda soyut düşünce anlatımını St. Thomas Aquinas’ın benzetimi destekler niteliktedir. Mimarın aklındaki ev: ev fikri olarak belirlenebilir, çünkü sanatçı, gerçek evi aklında tasarlamış olduğu forma benzetmeye niyet etmiştir (Aquinas, 1945). Bu da tasarımın tanımı için bizi “düşüncelerin inşaası fikrine“ çıkarır. Tasarlama fiili zihnin soyut fikirlerinin ürünüdür. Buna karşın, Platon’un desteklediği, Rene Magritte’in ünlü “Bu bir pipo değildir” önermesinin hatırlanabileceği üzere, soyut fikirlerin reele dönüşmesi mükemmel değildir (Şekil 2.7). Michel Foucault, Rene Magritte’nin tablosu ile karşılaşması üzerine, tablo ile aynı isimde bir kitap yazar ve bir ressam ile bir düşünür aynı düşüncede, dil felsefesinde buluşurlar. Kelimeler ve işaret ettikleri nesneler arasında bir ilişki var mıdır? Foucault Kelimeler ve Şeyler kitabında “..gördüğümüz şeyleri istediğimiz kadar anlatalım, görünen şey hiçbir zaman söylenen şeyin içine sığmaz..” der (Foucault, 2001, s.36). Magritte de görüntü

değil göstergelerin önemi olduğunu düşünen bir ressamdır ve "Bu bir pipo değildir" tablosunda, tablonun gördüğümüz şeyleri söylemede yetersizliğini vurgulamaya çalışmıştır. Buna göre, tabloda çizilmiş piponun hiçbir zaman gerçek bir pipo hiçbir mutlaklığı üzerine, Magritte'nin tablosu için "pipo değil piponun bir benzetimidir" ifadesi doğru olacaktır (Url-12-13).

Şekil 2. 7: Ceci n'est pas une pipe (Bu bir pipo değildir) - Rene Magriette, erişim <https://www.wikiart.org/en/rene-magritte/the-treachery-of-images-this-is-not-a-pipe-1948>.

Dil ve felsefedeki soyut ve reel ilişkisi, tasarım için tasarım sürecinin opaklığı üzerinden okunabilir. Platon'a göre düşüncenin kusurlu sonucu olan tasarımda, kusurun sebebi, zihnin içindekilerin dışarıya tam ve kusursuz çıkamamasından kaynaklandığını düşünebiliriz. Başka bir ifade ile tasarım sürecinin şeffaflığı azaldıkça, bilinmeyenler, tasarımcı içgüdüsü ve düşüncelerinin gizliliği arttıkça, tasarımcının kara kutusu açılmadıkça, sonuç üzerinde soyutluğun arttığını söylemek yanlış bir ifade olmayacaktır. Matematikçi, şair, bilim insanı Piet Hein'in şu sözleri önemlidir; "Sanat, çözülmeden önce formüle edilemeyecek bir problem çözmedir. Problemi şekillendirmek cevabın bir parçasıdır" (Url-14-15). 1944 yılında çalışma metodolojisini bu cümlesi ile ifade eden Hein, bugün aslen işlemsel tasarım ile irdelediğimiz noktalara değinmektedir. İşlemsel tasarım ile tasarım sürecinin aslen formalleştirilmesi ile tasarımın algoritmik tanımı yapılarak, nihai değerlerden uzak, ilişkiler ağı ile tasarım probleminin çözümlenmektedir. Süreçte formüllenen, Hein'nin altını çizdiği gibi problemin kendisi, tasarım girdileri ve problemin çözümüne yönelik bir algoritmadır. William Mitchell'ın da savunduğu üzere, işlemsel tasarımda ilk yapılan, girdileri tanımlayacak formüllemenin yapılmasıdır (Mitchell, 1990). Sean Ahlquist ve Achim Menges Computational Design Thinking kitabında işlemsel – hesaplamalı tasarımın temelinin, "..form ve matematiksel kurallarla inşa edilme sisteminin anlaşılması"na dayandığını savunur (Ahlquist ve Menges, 2011, s.16). Bu

yorumda öne çıkan önceden de dile getirildiği üzere "süreç" ve "sistem" kavramlarıdır ve bu kavramlar işlemsel tasarımın özünü oluşturmaktadır. "Çözüm için, içgüdüsel araştırmalara dayanması yerine, işlemsel-hesaplamalı tasarım, "adım adım bir süreç tekniği", kurallar ve girdilere teslim olmuş bir sonuçtur. Tasarım sürecinin bu şekilde düşünülmesi, "kurallara dayalı sistem", "algoritmik düşünceyi" işaret eder" (Jabi, 2013, s.22). Sürecin adım adım kurallara dayalı bir sistem içinde çözümlenmesi, her adımda sürecin şeffaflığını arttıracak ve nihai tasarım ile tasarımcının zihninde oluşmuş hali arasındaki mesafe azalacaktır.

Paul Coates tasarım yaklaşımının, büyük oranda karmaşık işlemin, en iyi şekilde ele alındığı çözüm süreci fikri ile ilişkili olduğunu, bunun da en iyi işlemsel tasarım ile uyuşmakta olduğunu savunur (Coates, 2010). Tasarım, büyük ölçekten küçük ölçeğe birbirleri ile çalışan sistemler, ilişkiler ağıdır. Bir prizmayı oluşturan yüzeyler, yüzelerini oluşturan çizgiler, çizgileri oluşturan noktalar kümesi ile bir organizmayı oluşturan sistemler, sistemleri oluşturan organlar, organları oluşturan dokular, dokuları oluşturan hücreler arasındaki fark ölçektir, her ikisi için de ilişki zinciri dizisi söz konusudur demek yanlış bir ifade olmamalıdır. Eğer mimarlar sistem tasarımcıları iseler, düşüncelerini geliştirmek için algoritmik olarak düşünmelidirler (Coates, 2010). Algoritmik düşünce ile karmaşık tasarım problemi, işlemsel – hesaplamalı tasarımda, tasarım ürününün en küçük yapısı olan noktadan başlayarak adım adım çözümlenmektedir. "İşlem – hesap, formu anlama ve algılamada derin bir etkiye sahiptir. Formu "idrak etmeden" formun "oluşumunu çözümlemeye" geçiş vardır" (Ahlquist ve Menges,2011, s.16). Oluşumun çözümlenmesine geçiş, aslen tasarımın aklına geçiştir; Ahlquist ve Menges işlemsel – hesaplamalı tasarımın, "form oluşumu ve tasarım sürecini çözümleme" yaklaşımına vurgu yaparken, bu ifadenin aslen tasarımın aklına dönüşü ifade ettiğini söylemek gerekir. Aklın çözümlendiği süreçte, "..sonuç ürününün modellenmesi yerine modelleme sürecinin kendisi tanımlanmaya başlanmıştır. Süreçte birbirini izleyen yönergeler, "algoritma", ile değişkenler dizisi, "girdi"ler, sonuç "çıktı"sını üretir. Girdiler değiştirilerek farklı çıktılar türetilir" (Scheurer ve Stehlig, 2011, s.75). Bu tarifte bir süreç tanımının yapılmasının yanında bu süreç, bir keşif eylemidir. "Form artık yapılan değil bir dizi kurala veya başka bir ifade ile algoritmaya bağlı, genel olarak dijital, ancak aynı zamanda fiziksel araç ve teknolojilerle bulunandır. Kurallar belirlenir, sürecin bütünü bu kuralları takiptedir" (Agkathidis, 2015, s.17). Sürece, akla hükmetmek, aklı yönetmek, sonucun tamamen

kontrolünü beraberinde getirir. Bu yeni durum tasarımda sonucun değil sürecin önemini arttırırken, tasarım için değerlendirmenin de değiştiği söylenebilir. Fabian Scheurer ve Hanno Stehling, tasarım yargılarındaki değişimi şu sözlerle ifade eder:

> Düşünüyoruz ki, yalnızca sonuç ürününün incelenmesi yerine sonsuz varyasyon türetilebilen sürecin niteliğinin tartışılması gereken bir zaman. Değişkenler dünyası içinde kaybolmak istemiyorsak, makinenin değil bizim tasarladığımız algoritma makinelerinin niteliğini anlamaya ihtiyacımız var. Modelin değişkenleri nelerdir? Algoritma nasıl tasarlanır, kurallar nasıl tanımlanır? Tasarımın niteliği ve parçaların birbirini hesap edebilmesi için ölçülebilir, eniyileştirilebilir nedir? Algoritmanın niteliğini ne tanımlar ve bu sonuç çıktısına nasıl yansır? Son olarak, dijital araçlarla değil, projeye işlenmiş insan akılları ile, kompleks mimari yapı ile, nasıl iletişim kurabiliriz? Zira "tasarlamak", kararları çizmek ve sorumluluk almaktır, araçlara görev devretmek değildir. Buna ancak kurallı tanımlara bağlı olan, kendi kendine kuralcı olan, algoritmik tasarım ile engel olunabilir. (Scheurer ve Stehling, 2011, s.79.).

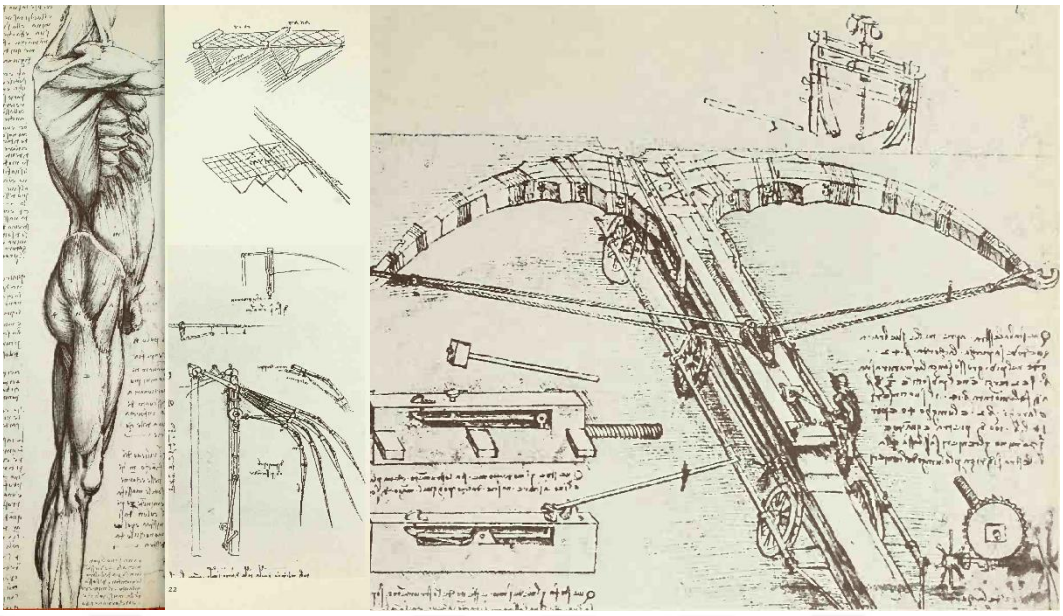
İşlemsel tasarım, algoritmalar ile düzenlenen ilişkiler ağının matematiksel dilde kurallarla oluşturulması sayesinde tasarım sürecini saydamlaştırarak, sonuç çıktısındaki soyutluğu azaltan açık bir "tasarım" tarifi yapmaktadır (Şekil 2.8).

Şekil 2. 8: Kurallı geometri (yazar tarafından yorumlanmıştır).

Tasarım için algoritmalar ile inşa edilen, William Mitchell'ın tanımlaması ile "tasarım dünyası", tasarım ile, deneyimleyerek, iletişime geçme ortamı sunmaktadır (Peters, 2013). Aslen R. Aish, F. Scheurer, W. Mitchell, P. Coates gibi bu alanda çalışmalar yapan, bahsi geçmiş tüm isimlerin de savunduğu üzere işlemsel – hesaplamalı düşüncenin özü algoritmik olarak düşünmektir. Bu dijital bir süreç değil, aksine insan zihninde yaratılışından gelen, yanlızca tasarımda değil insanla ilişkisi olan tüm alanlarda karşımıza çıkan bir düşünce yapısıdır. İlk kez programlanabilir bilgisayar

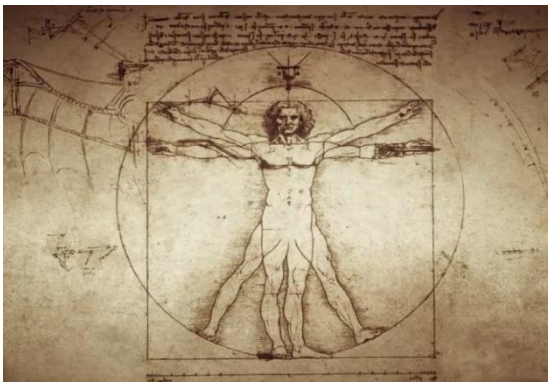
fikrini ortaya atan Charles Babbage ve ilk bilgisayar programcısı kabul edilen Ada Lovalace'ın icat ederek, dijital dünyanın temellerini attıkları, Difference Engine'de de karşımıza çıkmaktadır. Algoritmik düşünce insan hayatına ilk kez dijital araçlarla girmemiştir, aksine araçlar insanın doğasında olan bu düşünce sistemi ile ortaya çıkarılmıştır. Ancak, Paul Coates'in savuduğu üzere, algoritmik düşünce mimarlar tarafından değil mühendisler ve bilim insanları tarafından keşfedilmiştir. Bugün algoritmik düşünceyi bir tasarım metodolojisi olarak tartışırken, tasarlama sürecinin dijitalleşmesi yönünde değil, tasarımda işlemsel düşünceye uzaklığın eritilmeye çalışıldığı bir düşünce sisteminden söz edildiğinin farkında olunması gerekmektedir (Coates, 2010). Tasarımda işlemsel düşüncenin dijital araçlar ile oluşmadığına ispat gösterilecek pek çok önemli isim vardır. Antonio Gaudi'den Mimar Sinan'a, Leonardo Da Vinci'den, İlhan Koman, Naum Gabo'ya, Maurits Cornelis Escher'den, Piet Mondrian'a sayısız örneklerin sahiplerinin eserlerinde, dijital bir aracın zekasının değil, kendi matematik akıllarının (analitik zekalarının) yansıması gözlenmektedir.

1452 – 1519 yılları arasında yaşamış Leonardo Da Vinci, İtalyan mimar, mühendis, mucid, matematikçi, anatomist, ressam ve müzisyendir. Çalışmalarına yansıyan en önemli yaklaşımı, anatomiden, fiziğe, inceleme alanlarında daima sistemlerin çalışma prensiplerini incelemesi ve sistemler arası benzerlikler kurgulayabilmesidir (Gibbs - Smith, 1978). "Hiçbir araştırma matematik ispattan geçmedikten sonra ilim adını almaya lâyık olamaz" sözlerinin sahibi Da Vinci, bütün bilgilerin temelinin deneye dayalı olduğunu ve deneyden gelmeyen bilginin eksik olduğunu savunmuştur. Kadavralar üzerinde, sinirler, tendonlar, kaslar, iskelet sistemi ile ilgili incelemeler yaparak hareket, ağırlık, kuvvet prensibine dayalı deneyler yapmış, anatomi alanında keşfettikleri ile fizik alanında çalışmalar yapmış, hareketli makineler tasarlamış, kuşların kanat düzeneğini temel alan uçan makineler icat etmiş ve daha pekçok alanda keşfettiği herşeyi resmederek bugün farklı müzelerde ve koleksiyoncuların ellerinde olan yaklaşık 13 ciltlik 5000 sayfa not bırakmıştır (Şekil 2.9) (Url-16).

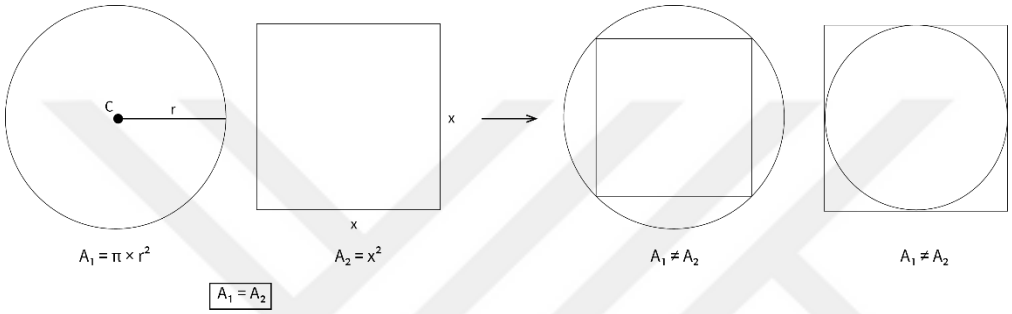


Şekil 2. 9: Leonardo Da Vinci eskizleri, Gibbs – Smith, C., (1978)., The Inventions Of Leonardo Da Vinci kitabından alınmıştır.

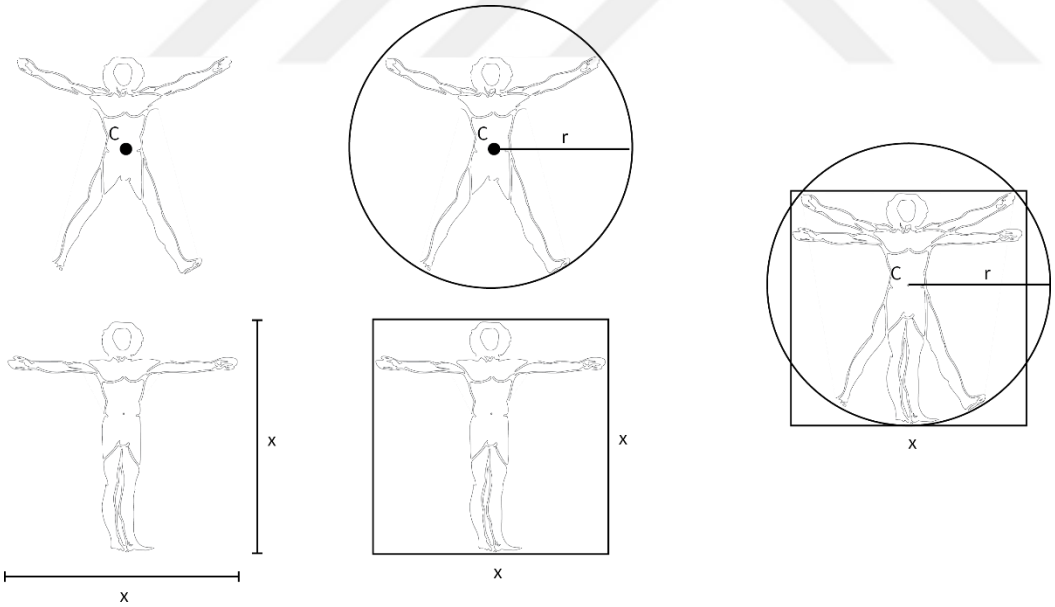
İnsan vücudunda oransal ilişkileri gösteren Da Vinci'nin Vitruvian man eskizi, Rönesans döneminin sembolü olmuştur (Şekil 2.10). Daha sonra Le Corbusier tarafından Modulor olarak yeniden yorumlanacak Vitruvian man geometrinin insan odaklı tanımlamasıdır (Ceccato ve diğ, 2010). Da Vinci eskizini Vitruvius'un De Architectura kitabında anlatıkları üzerine yapmıştır. Vitruvius'un bakış açısından etkilendikten sonra kendi Vitruvian man önermesi oluşmuştur (Berloquin, 2008). Alanları eşit çember ve karenin nasıl oluşturulacağı ilk kez antik dünyada ortaya atılan ve "çemberin karelenmesi" olarak bilinen problemdir ve Pi sayısının doğası gereği problem çözümsüzdür (Şekil 2.11). Rönesans döneminde Da Vinci Vitruvius'un eserinden etkilenerek çember ve karenin ortasına insan figürü yerleştirip "çemberin karelenmesi" bilinmezini çözümlemiştir. Vitruvius insan göbeğinin vücudun tam ortası olduğunu, iki yana açılmış kol genişliğinin de insan boyu ile mükemmele yakın eşit olduğunu iddia eder ve Da Vinci eskizinde insan göbeğini çemberin merkezi olarak, insan boyunu da karenin bir kenarı olarak kullanır (Url-17-18) (Berloquin, 2008). Çember ve kare içine çizilmiş basit bir insan figürünün ardında matematiksel bir oranlama, hesap ve çözümleme olduğu görülmektedir (Şekil 2.12). Sayılarla, uzunluk ve genişlikleri ilişkilendiren kuralların haritalaması, tasarımcıların basit algoritması, Vitruvian man eskizi ile yapılmıştır (Gönenç Sorguç ve Arslan Selçuk, 2013).



Şekil 2. 10: Vitruvian Man – Leonardo Da Vinci, erişim <https://handheldcamera12.wordpress.com/2013/02/14/vitruvian-man/>.



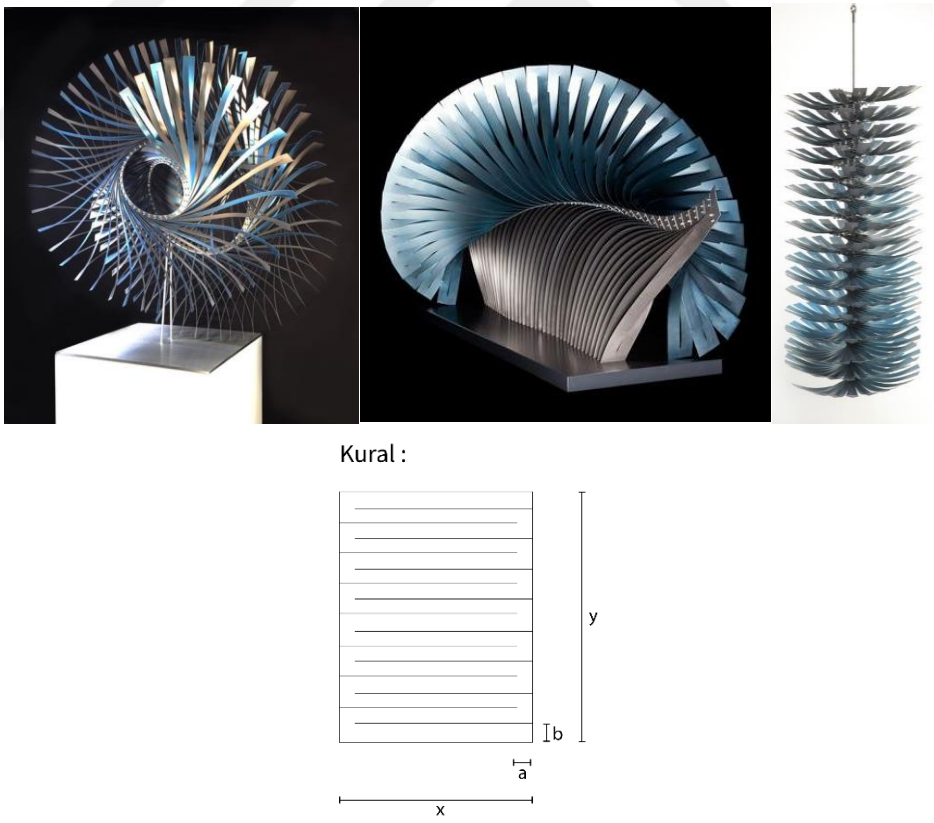
Şekil 2. 11: "çemberin karelenmesi" sorunsalı.



Şekil 2. 12: Vitruvian Man – Leonardo Da Vinci "çemberin karelenmesi".

İlhan Koman 1921 – 1986 yılları arasında yaşamış heykel sanatçısıdır. Türk Da Vinci'si olarak kendisinden bahsedildiğine rastlamak mümkündür. Matematiğe özel ilgisi olan Koman, yaptığı heykeler için kendi geliştirdiği matematik formülleri ile formlar oluşturmuştur. Geliştirdiği bu formülleri bazı heykelleri için On My Approach

To Making Nonfigurative Static and Kinetic Sculpture makalesinde anlatmıştır. İlhan Koman işlerinden "embriyonik" olarak bahseder ve niyeti, her parçanın yeni fikirler üretme ve aynı türün daha gelişmiş örneklerini üretmede kullanılabilecek bilgiyi içerdiğinin altını çizmektir (Koman ve Ribeyrolles, 1979). Ancak bu yaklaşımın durağan bir yönde ilerlemek olarak düşünülmemesi gerektiğini dile getirir. "Asıl ilgilendiğim, deneyimlerimde ortaya çıkan sayısal problemler ve bu problemleri çözme girişimidir" der (Koman ve Ribeyrolles, 1979, s.1). Form üretiminin arkasındaki matematiksel düşünce ve Koman'ın form arayışındaki kavramsal yaklaşımı, işlemsel tasarım ile aynı bağlamdadır (Gürsoy ve Özkar, 2015). 1970'lerde Leonardo Da Vinci, Alman matematikçi Möbius'un sonsuzluk işareti gibi ilgi alanları olan Koman, bu ilgisinin yansıması olarak görülebilecek serilerine başlamıştır. Hyperforms - Polyhedra - Derivatives, Infinity-1 ve Pi serileri bu döneme aittir (Şekil 2.13). Şekil 2.13'de görülebilecek bu serilerdeki yapıtlarında aynı kuralın farklı türevleri ile oluşmuş formlar üretir ve kurallara bağlı olması sebebi ile formlar yeniden üretilebilir özelliktedir (Beşlioğlu, 2011).

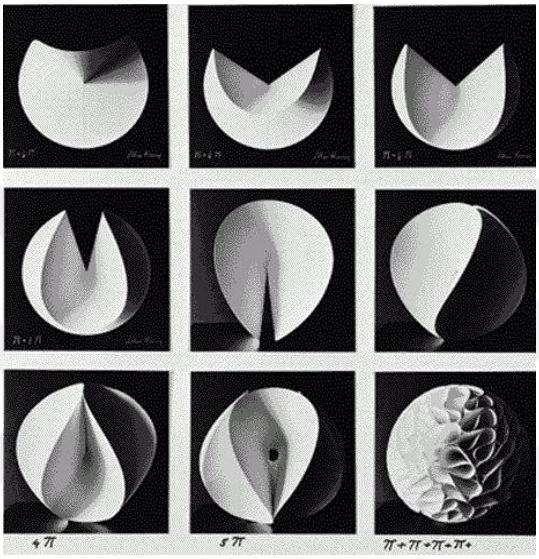


Şekil 2. 13: "Whirlpool" - "To Infinity" - "İsimsiz", erişim Egeran galeri.

İşlemsel tasarım, tek bir ürün çıktısı yerine süreç tasarımı ile form olasılıkları üretme yaklaşımındadır ve İlhan Koman'ın ürettiği matematiksel formüller ile formülün farklı

değerleri ile keşfettiği formlar bu bağlamda, Koman'ın dijital araçlar kullanmasa da işlemsel ve ilişkisel düşüncesinin sonucudur. Pi serisinde İlhan Koman yarıçaplarını sabit tutup, Pi katlarını arttırarak elde ettiği dairesel yüzeyler ile formlar oluşturmuştur (Şekil 2.14) ve yaklaşımını şu sözleri ile aktarmıştır:

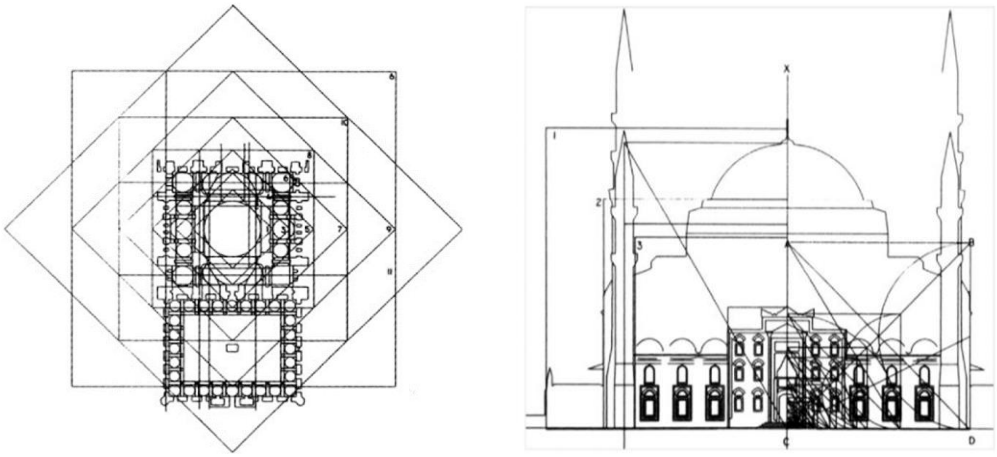
> 1 Pi'den daha fazla Pi ile yüzey nasıl oluşabilir sorusunu kendime sordum. Bu seri de bunun cevabı. 2 boyutlu bir çembere, çember kesiti eklenirse, 3 boyutlu olur. Kesitin açısı arttırılırsa, daha fazla sayıda ve daha büyük eğriler oluşur... 3Pi, 4Pi, ve fazlası için yeni formlar oluşacaktır... Çok sayıda Pi ler küre gibi bir form oluşturacaktır. (Beşlioğlu, 2011, s.13).



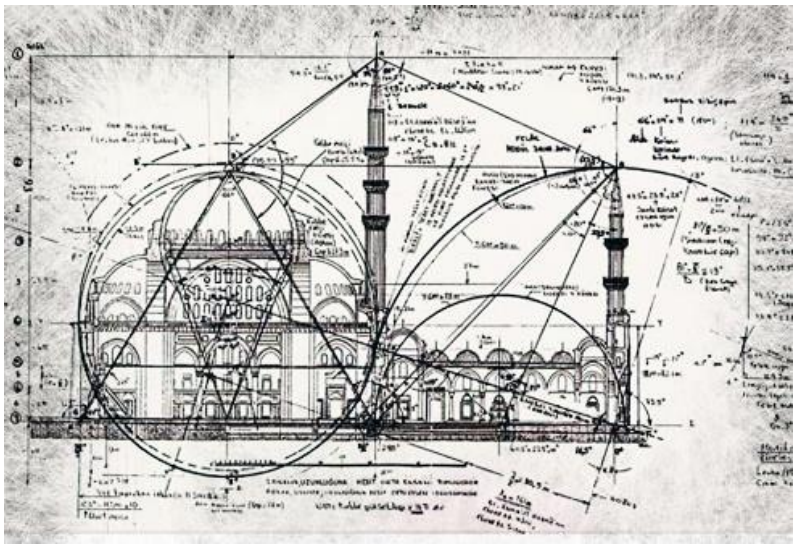
Şekil 2. 14: Pi Serisi - İlhan Koman, erişim <http://designspiration.net/image/2433600081671/>.

Mimar Sinan 1489 – 1588 yılları arasında yaşamış, 99 yıllık ömründe 400'ü aşkın yapının mimarlığını üstlenmiş, Osmanlı İmparatorluğu'nun en güçlü olduğu çağda 50 sene imparatorluğun başmimarı olarak görev yapmıştır. Bina teknolojisi ve estetik kaygılar açısından Osmanlı'da en yüksek seviye olan "Klasik Osmanlı Dönemi" üslubu, dönemin en önemli yapılarının mimarı Sinan'ın eserleri ile oluşmuş, dönem "Mimar Sinan Dönemi" olarak da adlandırılmıştır (Sağdıç, 2015). Mimar Sinan her yapısında adım adım yapısal problemleri çözümlemekte ve problemlerde ilerleme kaydetmektedir, bu sebeple onun eserlerinin tamamının süreç olarak incelenmesi gerekmektedir (Yazar, 2003). Bu dönemler Mimar Sinan'ın tasarladığı üç büyük eser olan Şehzade, Süleymaniye ve Selimiye külliyelerinin inşaasıyla Sinan'ın kubbeli yapı problemini geliştirmesi üzerinden ayrılıp, uzmanlar bu dönemleri "çıraklık", "kalfalık" ve "ustalık" olarak isimlendirmektedir (Benian, 2011). Mimar Sinan'da görülen bu dönemler, süreç tasarımı olarak yaklaştığımız işlemsel tasarım alanı için

bile, birden çok tasarım sürecinin iç içe geçtiği, tasarım problemi odaklı yaklaşımın üst mertebe bir üslup içerdiğini söylemek yanlış olmamalıdır. Aslen işlemsel tasarımda da bir tasarım problemi için geliştirilen süreç modeli, farklı tasarım süreçlerinin alt yapısını oluşturmakta ve sonuç ürünü modellemesi yanında süreç modellemesi sonsuz bir süreklilik kazanmaktadır. Bu yaklaşımın yüzyıllar önce Sinan'ın mimari anlayışında mevcut olduğunu görmekteyiz. "Zamanının Öklid'i" Mimar Sinan geometriye dayanarak mimarlığı anlayıp bu anlayışla form sistemi kurar (Kuban, 1988). Özellikle Mimar Sinan ile ilgili araştırmalar yapan fizikçi John Freely ve mimar, mühendis Augusto Romano Burelli "Öklid geometrisi Sinan'ın özel kompozisyonlarında ve inşa kurallarında büyük etkiye sahiptir" der (Sağdıç, 2015). Mimar Sinan'ın işlerinde en önemli eleman kubbedir ve strüktürün tamamının merkez noktasını oluşturur. Strüktürün bütünü kubbeyi taşımak üzere şekillenir. Elemanlar arası ölçü ve ilişkiler insan vücudunun ölçülerine, modüler oranlamalara, geometrik ilişkilere dayanır (Bilgin, 2006). Mihrimah Sultan, Şehzade ve Rüstem Paşa camilerinde Mimar Sinan, yarıçapı 3 arşın (68 santimetre) olan daire içine yerleştirilmiş sekizgenden oluşan modüler sistem kullanmış, ancak bu ana modülün ölçülerinin 3 katı 9 arşın ile çalışarak yapı yüksekliği, kubbe saçaklarını da oranlama ile oluşturmuştur. Süleymaniye camisinde altın oran kullanarak yapıyı tasarlamıştır (Sağdıç, 2015). Bu yapılar Mimar Sinan'ın yanlızca birkaç eseri olmasının yanında, bütün yapılarında geometri ve matematik kullandığı ve tüm mekan organizasyonunu matematiksel kural ve oranlamalarla çözümlediği anlaşılmaktadır (Şekil 2.15-2.16).



Şekil 2. 15: Süleymaniye Camii plan – kesit geometrik şeması, Çizimler: Zafer Sağdıç.



Şekil 2. 16: Şehzade Camii geometrik şeması, erişim https://murattuzcu.me/tag/sehzade-camii/.

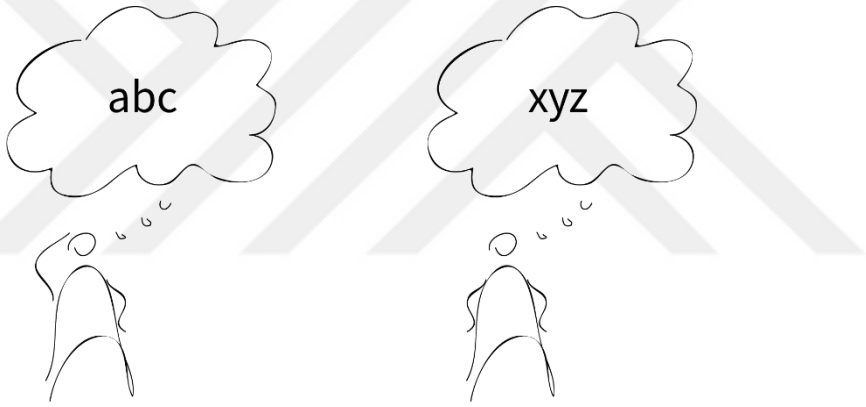
Leonardo Da Vinci, İlhan Koman, Mimar Sinan farklı dönemlerde, farklı alanlarda çalışmış, çalışmaları dönemlerinin sınırlarını aşmış isimlerdir. Bu isimlerin ortak paydası işlemsel zekaları, sistematik düşünce yapılarıdır. Eserleri, matematiksel kural ve formüllerin somutlaşmış hali, başka bir ifade ile matematiksel ifadelerin yansımalarıdır. Bugün kazandırdıkları eserler hala ileri düzey bir aklın göstergesi olmakla birlikte, işlemsel tasarımın matematik ve tasarımcı arasındaki perdeyi kaldırması ile kompleks formları anlama ve oluşturmanın yakınlığı gittikçe artmaktadır. Kurallı geometriler ve matematiksel sistemler kurmak bir dehanın ürünü olmaktan çok, tabana yayılan bir yaklaşım olmaya başlamaktadır. Bugün geldiğimiz noktada, mimarlar matematik ile ilişkisini sorgulamadan, hiç kullanmadıkları kadar matematik kullanmaya başlamış durumdadır (Picon, 2011).

3. LİSAN

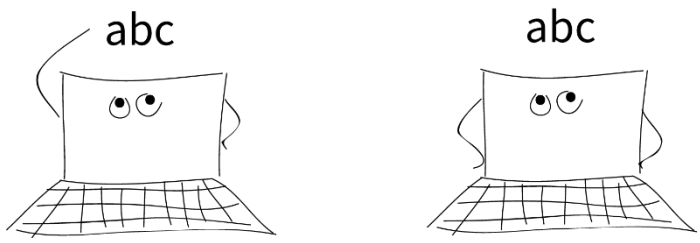
“İnsanlar olmadan dil olabilir mi? Ya da daha iyisi, insan dışında diller olabilir mi ve bizler insanlar olarak bu insan dışı dilleri ne derece kavrayabiliriz?” (Galloway ve Thacker, 2006, s.27).

Tüm sistemlerin özü, aklı okumanın ve anlamanın ötesinde, aslen aklı şekillendiren bir faktör vardır. Bu da dildir. Dil yüzyıllardır gelişen, dilbilgisi ve dilbilimi uzmanları tarafından araştırılmakta olan ve aslen başlangıcını, kullanılan ilk dili, dili öğrenmede beynin çalışma yöntemini henüz tam olarak bilemediğimiz iletişim aracıdır. Bu sebepledir ki, dil ile ilgili düşünürlerin farklı fikirleri vardır. Dil ve düşünme ilişkisine ilk kez Platon dikkat çekerek “İnsanın kendi kendisiyle içinden yaptığı konuşmadır” derken, herşeyin temelinin dil olduğunu savunan Sofistler, birşeyin dilde ifadesinin olmamasının onun var olmaması manasına geldiğini savunur. Aristoteles dilin insanları hayvanlardan ayırdığını düşünürken aynı fikirde olan Descartes, hayvanlarda dil yetisinin hiç olmadığını, öğrenebildikleri sınırlı kelimenin onların ancak korku, umut, sevinç hareketlerinden başka birşey olmadığını ileri sürer. Leibniz ise, dil aklın aynasıdır savı ile karşımıza çıkar (Url-19). Temel olarak gelişen ve tartışılan tüm fikirler dil ile düşünce arasındaki ilişki üzerine olmuş, insanı diğer canlılardan üstün yapan akıl seviyesinin yanında, dil için de aynı şeyler düşünülmüştür. Benjamin Lee Whorf’un dediği gibi “Dil, düşünme tarzımızı şekillendirir ve neyi düşünebileceğimizi belirler” (Url-19). Bu sebeple, akıl gibi insan dilinin de diğer dillerden üstün olduğu yönünde düşünceler gelişmiştir. Doğal dillerin yapay dillerden en büyük farkı da düşünce ile olan bu ilişkisidir. Noam Chomsky, Language and Mind (Dil ve Zihin) kitabında, Port Royal kuramı üzerinden bir anlatım yapmaktadır. Port-Royal Grammar 1660 yılına ait, felsefi dilbilgisini başlatan yapıt olarak kabul edilebilmekte olup, Port-Royal kuramı tümce içinde karmaşık fikrin bir dizi sözöbeğine karşılık gelmesidir (Chomsky, 2001). Buna örnek olarak Chomsky “a wise man is honest” (“bilge insan dürüsttür”) tümcesini inceler. Port-Royal kuramına göre “bilge insan dürüsttür” “yüzey yapısı” olarak adlandırılır. Bununla birlikte “a man is wise” (“insan akıllıdır”) ile “a man is honest” (“insan dürüsttür”) önermeleri “derin yapıyı” oluşturarak

karmaşık fikri içerir, ve aynı zamanda "bilge insan dürüsttür" cümlesi ile bu önermeler direkt olarak söylenmez (Chomsky, 2001, s.35). Dolayısı ile yüzey yapısı içerisinde derin yapılar mevcut olup, derin yapı yüzey yapısına bir takım zihinsel işlemler ile bağlıdır. Anlam içerisinde anlam barındırma olarak yorumlayabileceğimiz bu durum, ancak doğal dillerde mevcuttur. Başka bir ifade ile bir bilgisayar, programlama dilinde yazılmış bir metinden alt metinler çıkaramaz, yorum yapamaz (Şekil 3.1-3.2). Bu duruma bir başka örnek de Chomsky'nin dil öğrenme süreci ile ilgili düşünceleridir. Dil öğrenmenin taklit ile değil, "dil kullanımının yaratıcı yanı" olarak nitelendirdiği, olağan dil kullanımı ile gerçekleştiğini ileri sürer. Chomsky, söylediğimiz birçok şeyin bütünüyle yeni olduğu, daha önce duyduğumuz hiçbir şeyin yinelenmesi olmadığı, ve geçmişte duyduğumuz hiçbir söyleme yapı bakımından benzer olmadığını düşünmektedir (Chomsky, 2001, s.29). Bu sav da, dili öğrenme dönemi itibari ile, bir yorumlama ve süzgeçleme mekanizmasının devrede olduğunu göstermektedir.



Şekil 3.1: Doğal dillerde iletişim (yazar tarafından yorumlanmıştır).



Şekil 3.2: Yapay dillerde iletişim (yazar tarafından yorumlanmıştır).

Doğal dillerin aksine ikincil anlamların, yorumlamanın, düşünmenin olmadığı yapay dillerin temeli, 1948'de Claude Elwood Shannon'un geliştirdiği "matematiksel iletişim" kuramına dayanmaktadır. Shannon, iletişim sisteminde mesaj iletiminin, kullanışlı, gerçek değerlerde ve matematiksel olması açısından logaritmik bir fonksiyon ile yapılması gerektiğini savunur (Shannon, 1948). İnsan iletişiminin

rastgelelik ile istatiksel bağımlılıkların bir karışımı olduğunu düşünmekte olup, matematiksel iletişim modelini telgraf iletişimi üzerinden geliştirmiştir. İletilen harflerin bir bakıma kendinden önceki harflere bağlı olduğu savından yola çıkarak a, b ve c harflerinden oluşan bir alfabe modeli üzerinden örnek bir metne en yakın metni oluşturma üzerine çalışmıştır (Shannon, 1948). Bu modelde, 1. adımda rastgele 3 harfin olduğu bir havuzdan seçim yaptığını varsaymış, 2. adımda iki harften oluşan gruplar arasından seçim yapmış, 3. adımda ise gruplamasını 3 harften oluşan gruplar ile yapmıştır. Her adımda orjinal metne daha fazla yaklaşmış, daha sonra bu örnek modeli gerçek bir İngilizce metin üzerinde uygulamış, ilk önce harfler, harf çiftleri ve harf üçlüleri denemesinin ardından, modeli kelimelerle tekrarlamış ve herhangi bir ingilizce metne olan benzerliğin her seviyede ciddi şekilde arttığı sonucuna ulaşmıştır (Url-20). Sonuçta, iletilecek "bilgi" miktarının makineye eklenmesi gerektiğini fark etmiş ve bunu "entropi" kavramı olarak adlandırmıştır. Kurallara dayalı bu iletişim modeli, bilgisayarlar ile iletişimimizi sağlayan yapay dillerin tanımının başlangıcı olmuş, doğal dillerin bilinmeyen sözdizimleri, doğal gelişim ve değişimlere tabi sözlüklerinin yanında, yapay dillerin açık sözdizimleri ve iyi tanımlanmış sözlükleri olmuştur (Coates,2010).

3.1 Bilgi-sayar

"Sonucun büsbütün kontrolünde olmak isteyen tasarımcı, sürecin kontrolünde olmalıdır. Sürecin kontrolünde olmak için tasarımcı, araçların kontrolünde olmalıdır" Robert Aish (Burry, 2011, s.67'de atıfta bulunulduğu gibi).

İnsanın "sayma" isteği önce rakamların icadı ile çözülmüş, ancak hızla gelişen dünyada sayımı yapılacak değerler hızla büyümüş, insanoğlu sayma üzerine geliştirdiği her icadında tarihte yeni bir dönüm noktasına gelmiştir. Modern bilgisayarın icadını sağlayan ilk gaye de sayma arzusudur. Ancak saymanın mekanikle buluşması kaçınılmazdı ve Ring Larnder'in desteklediği üzere tüm sıkıcı monoton sayısal hesaplama, mekaniğin sapmaz kesinliğine döndürülmüştü (Url-21).

Rakamların icadından sonra insan, sayma ve işlem yapabilmek için bir alet bulma isteğindedir ve bugün her alanda insan hayatına yerleşmiş bilgisayarın ilk habercisi, insanoğlunun ilk hesap makinesi, abaküsler bu amaçla uzun yıllar kullanılmıştır (Url-21). Ancak sayma yeteneğinden sonra, asıl ihtiyaç sayıların kaydedilmesi gerekliliğidir ve mevcut sistemler ancak tek bir hesabı çözebilecek durumdadır

(Williams, 1990). Michael R. Williams Computing Before Computers kitabında çoban ve koyun benzetimi ile şu örneği verir; çobanlar gündüz her bir koyun için çantalarına bir çakıl taşı koyarlar, gece topladıkları her koyun için çantalarına koydukları taşları çıkarırlar. Ertesi gün bu yeniden tekrarlanır. Bir önceki güne dair hiçbir bilgi kayıt edilemez ancak tek bir hesap yapılır (Williams, 1990). Her yeni işlem için yeni bir sistem kurmanın önüne geçebilecek yöntem, bir mekanizmanın kontrolünde işlem yapabilecek aletlerin keşfidir. Bu amaçla, 17. yüzyıl ile çarpma, bölme gibi aritmetik işlemleri hızlı ve doğru yapmak amacı ile analog ve dijital hesap makineleri, tablet makineleri icadı başlamıştır (Campbell-Kelly ve diğ, 2013). 1614 yılında İskoç matematikçi John Napier, logaritma yaklaşımı ile hesap makinelerine farklı bir anlam kazandırmış, 1620'de İngiliz matematikçi Edmund Gunter, trigonometri ile hesap takibi özelliği olan Gunter Scale'i geliştirmiş, William Oughtred sürgülü cetvel ile hesaplamalar yapmıştır. 1623'te Alman gökbilimci, matematikçi Wilhelm Schickard ilk hesap makinesi Calculating Clock'u bulmuş, daha sonra Leonardo Da Vinci makinenin prensiplerine göre bugün üretilebilecek, gelişmiş halinin eskizlerini yapmıştır. 1642'de Fransız filozof, matematikçi Blaise Pascal Pascaline ya da Arithmetic Machine olarak bilinen yanlızca toplama ve çıkarma yapabilen hesap makinesini icad etmiş, 1671'de Alman filozof ve matematikçi Gottfried Wilhelm von Leibniz Pascaline'ye çarpma özelliğini de ekleyerek Step Reckoner'ı tasarlamıştır. Endüstri devrimi ile her alanda mekanikleşmenin hayata girmesi, hesaplama alanında 1820'de Charles Xavier Thomas de Colmar'ın Arithmometer'i ile olmuştur (Url-21).

19. yüzyılın başlarında keşfedilmiş iki alet; Jacquard loom ile delikli kartlar ve iğneli müzik kutusu otomatikleşme ve programlama için önemli buluşlar olmuştur (Bromley, 1990). 1804 yılında sanayi devriminde büyük rolü olan ve bu tarihten sonra modern bilgisayar programlamanın kurucusu olacak delikli kartlar, Fransız dokumacı Joseph-Marie Jacquard'ın geliştirdiği, Jacquard loom dokuma makinesi ile ilk kez kullanılmıştır. Jacquard, karmaşık dokuma desenlerini basitleştirmek, dokumayı hatasız hale getirmek amacı ile yüzlerce delikli kart ile çalışan bir alet geliştirmiştir. Delikli kart üzerindeki her sıra piksellenmiş desen dizisini oluşturmakta, ipliğin bu deliklerden geçmesine izin vermektedir (Url-22). Bu sayede kartlara aktarılmış desen, daha sonra başka dokumalarda defalarca kullanılabilir, değiştirilebilir, uzatılabilir hale gelmiştir. Aslen insanın zihninde tasarladığı desen kartlar üzerindeki delikler olarak soyutlanmış, mevcut desenin elle tutulabilir kartlar haline dönüştürülmesi açısından

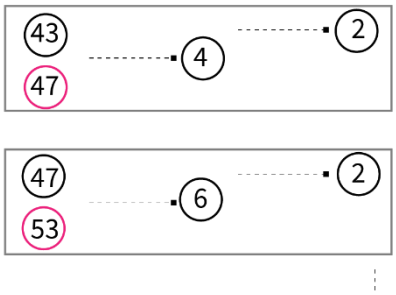
düşünüldüğünde tasarım, somut hale dönüşmüş, dokuma süreci bu sayede otomatikleştirilmiştir. Hesapları mekanize etme niyeti insanları, delinmiş bir kartın makineyi yönetmek için bir araç olarak kullanılabileceği, makinenin yaptığı işlem dizisinin kontrol edilebildiği, ve en önemlisi bir cihazın talimatlarının bir çeşit dil ile yönlendirilebileceği sonuçlarına, başka bir deyişle, makineyi programlama sonucuna ulaştırmıştır. Makine ve programlama, 20 yıl sonra Charles Babbage'ın ilk bilgisayarında kendini göstermiştir (Url-21).

Başarılı bir hesaplama makinesinin geliştirilmesi için, sayıların depolanması ve aritmetik olarak manipüle edilebilmesi, istenilen hesaplama sonucunu üretebilmek üzere aritmetik işlemler dizisinin kontrol edilebilmesi gibi mekanizmanın sağlaması gereken temel işlevler vardır (Bromley, 1990). Bu amaçla 19. yüzyılda otomatik hesaplama makinelerini geliştirmek için mekanizmalar ve fikirler geliştirilmiş ve bu dönemde bugünkü modern dijital bilgisayarların çalışma mantığı İngiliz matematikçi Charles Babbage tarafından ortaya atılmıştır (Bromley, 1990). Babbage müzik kutusu ve delikli kart fikirleri ile önce Difference Engine daha sonra Analytical Engine üzerine çalışmıştır (Randell, 1975). Fark metodu tekniği Charles Babbage'ın ilk otomatik hesap makinesinde kullandığı yöntemdir. Babbage'in icatlarının detaylı incelemelerini yapan Allan G. Bromley, Difference Engine'de kullanılmış fark metodunu şu şekilde anlatır: $T = x^2 + x + 41$ formülünden farklı x değerlerine göre türetilen T sonuçlarının birbirleri arasındaki fark, Δ, birinci fark değeri, birinci fark değerlerinin birbirlerinden çıkarılması ile elde edilen, Δ^2, ikinci fark değerleridir (Şekil 3.3).

x	T $x^2 + x + 41$	Δ $T^{(n)} - T^{(n-1)}$	Δ^2 $\Delta^{(n)} - \Delta^{(n-1)}$
0	41		
		2	
1	43		2
		4	
2	47		2
		6	
3	53		2
		8	
4	61		2
		10	
5	71		

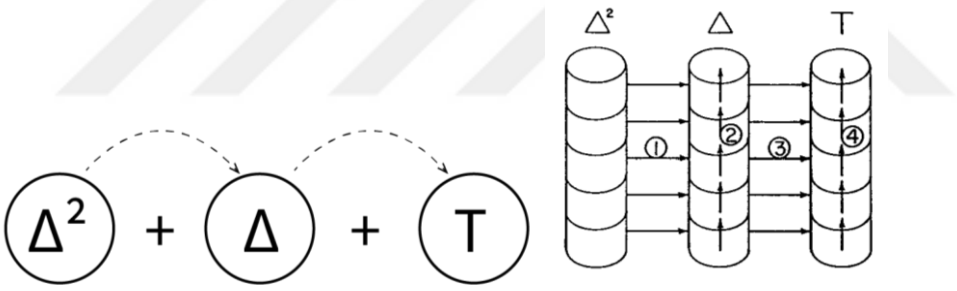
Şekil 3.3: Computation Before Computers kitabında yer alan Allan G. Bromley'in fark metodu şeması yazar tarafından uyarlanmıştır.

$\Delta + \Delta^2$ yeni bir fark değeri oluşturur ve bu değerin çizelgedeki T değerlerine eklenmesi bir başka T değeri sonucuna ulaşmamızı sağlar (Şekil 3.4).



Şekil 3.4: $T_1 + \Delta + \Delta^2 = T_2$.

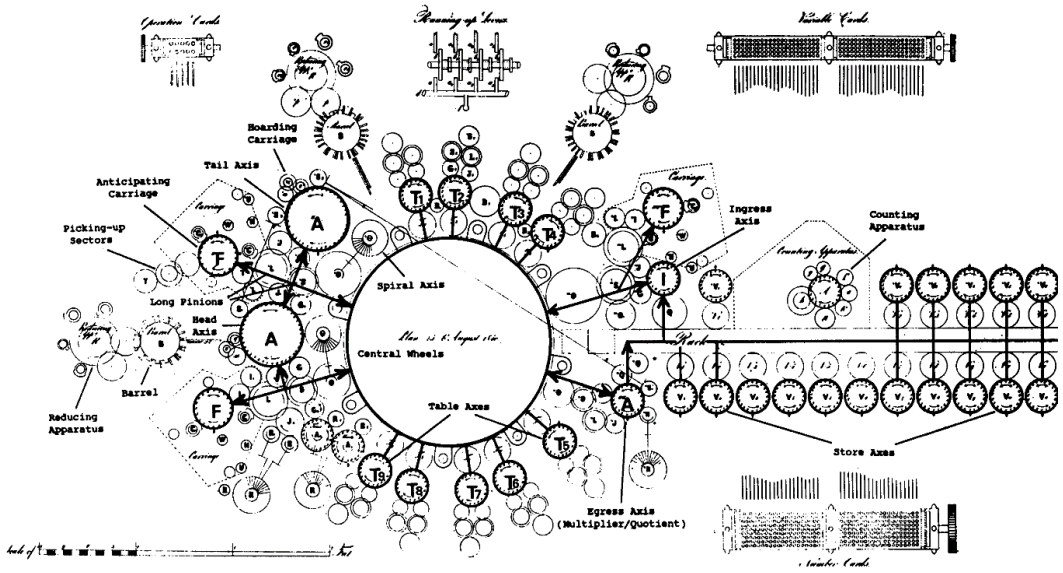
Burada anlatılmaya çalışılan basit matematiksel gerçeklik çok terimli bir fonksiyonun alt fonksiyonlara bölünebileceği ve bu aralıklarda oluşturulan değerler ile ana fonksiyondan türetilen sonuçlara yaklaşılabileceğini göstermektedir. Bunu alt çizelgeleme olarak adlandırılmış Charles Babbage, makinelerin de alt çizelgeleme yöntemi ile çalışabileceğini fark etmiştir (Bromley, 1990). Difference Engine de fark değerleri ile fonksiyon değerlerini türetme mantığı üzerine geliştirilmiştir (Şekil 3.5).



Şekil 3.5: Babbage'ın Difference Engine prensibi; fark metodu ile çizelgeleme (Bromley, 1990).

Charles Babbage fonksiyon işlemlerini gerçekleştirebilecek mekanizmasını, işlemdeki basamakları oluşturan akslara bağlı dişli çark sistemi ile kurgulamış, bir fonksiyona göre sonuçlar üretme işlemini otomatize edebilmiştir. Difference Engine matematik prensibi ile ve tekrarlı toplama yöntemi ile çalışmakta olup tek amaca yönelik bir makine olmuştur. Makineye girilen değişken değerlerin bir fonksiyona göre sonucuna ulaşılmaktadır. Fonksiyonun sonuçlarını tablolama açısından Difference Engine önemli bir adım olsa da matematiksel olarak henüz sınırlıdır. Campbell-Kelly'nin fikrine göre Babbage'ın başarısı bir zeka değil, zanaat başarısıdır (Swade,2003). Bunu doktora tezinde Doron David Swade, matematiksel bir keşif veya bir teori olmaktan çok bilgiyi işleme konusunda, dönemde bir gelişme olduğu şeklinde yorumlamıştır.

Babbage, 1830'larda Difference Engine'den farklı olarak genel kullanımı olabilecek, delikli kartlar ile programlanabilen, fonksiyonların farklı dizilerde uygulanabileceği Analytical Engine üzerine çalışmaya başlamıştır (Swade, 2003). Diffference Engine fark teorisini tekrarlı toplama işlemi ile yani tek işlem üzerinden gerçekleştirdiği için yetersizdir. Bu sebeple Babbage, ilk önce basamakları temsil eden eksenler üzerinde sayıları yukarı veya aşağı basma yeteneği ile makinenin, çarpma ve bölme işlemlerini kazanmasını sağlayacağını fark etmiş, 1834 sonlarına doğru Analytical Engine için temel planları oluşturmuştur ve mekanizmaya delikli kartları da ekleyerek evrensel bir hesap makinesinin tasarımını formüle etmiş ve yıllar içerisinde birden çok tasarım gerçekleştirmiştir (Bromley, 1990). Babbage, Analytical Engine'de sayılar için ayrı bir bölüm tasarlamış ve işlemi gerçekleştiren çark kısmına sayıları taşıyacak bir sistem tasarlamıştır (Şekil 3.6).



Şekil 3.6: Mekanizmanın işlevsel bölümlerinin ve arabağlantılarının mantıksal diyagramı; sol çarklı bölme, işlemin yapıldığı mekanizma iken sağ kısımda sayılar için geliştirilmiş bölüm görülmektedir (Bromley, 1990).

İşlem karmaşıklığını önlemek için ise, kısmi depolama özelliğini geliştirmiştir, bu uygulama elektronik bilgisayarlarda da aynı şekilde işlemektedir (Bromley, 1990). Daha basit bir ifade ile söylemek gerekirse, çok adımlı işlemlerde, kısmı depolamada ile ilk yapılan işlemler bekletilip, diğer işlemlerin sonuçlarına aktarım yapılmaktadır. Bu sayede Difference Engine'de bir işlem, temel fonksiyon iken, Analytical Engine'de ana fonksiyonun alt işlemleridir. Bu da ard arda işlemler yapabilme özelliği anlamına gelmektedir.

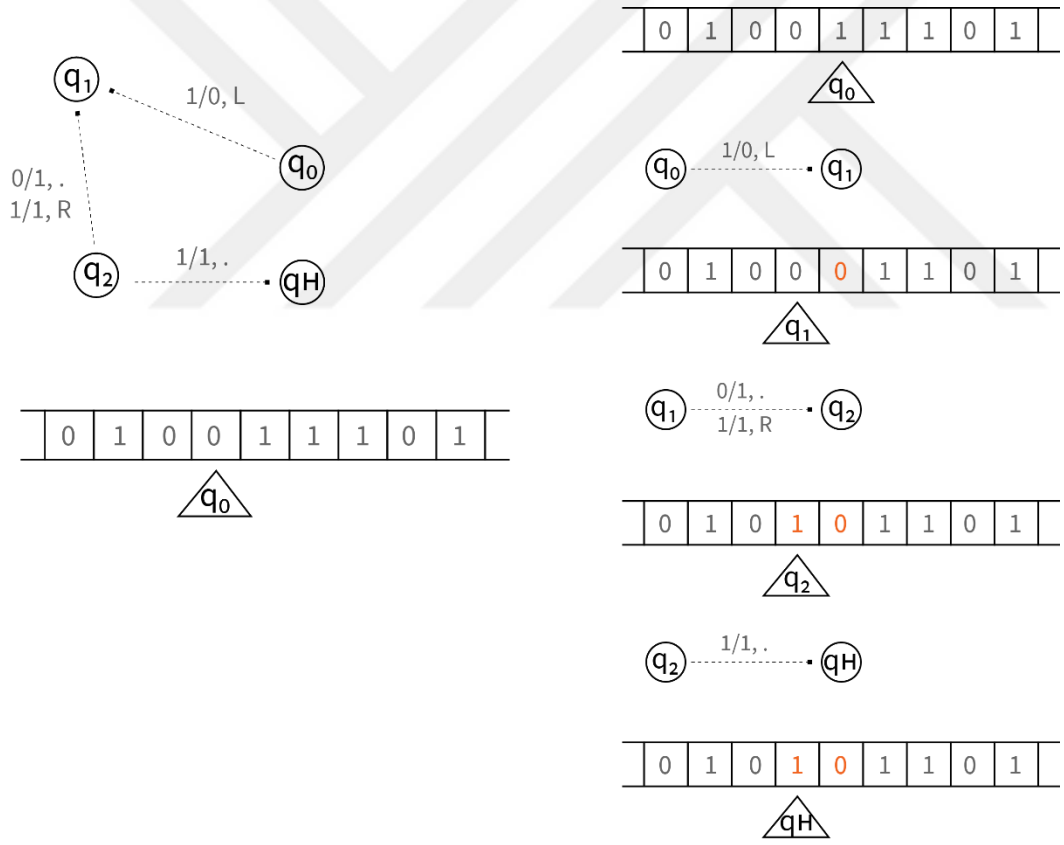
Charles Babbage'ın mekanik teknolojisi elektronik cihazlardan farklı olmasına rağmen, bugünkü modern bilgisayarın mantığına benzer tasarıma sahip, "store" (bellek), "mill" (merkezi işlem birimi) ve delikli kart okuyucusu ile modern bilgisayarın analog bulgularını taşımaktadır (Kim ve Toole, 1999). Analytical Engine'nin programlanması konusu ile ilgili ise fikir ayrılıkları vardır. Analytical Engine'nin programlamasına katkısı ile, ilk bilgisayar programcısı olarak İngiliz şair Lord Byron'ın kızı Ada Lovelace'ın bilinmesi, Allan G. Bromley tarafından bir kanıtın olmaması öne sürülerek kabul edilmemektedir. Bromley, Lovelace'ın katkılarının olduğunu ancak, Babbage'ın programlama kısmını kendisinin geliştirildiğini savunmaktadır (Bromley, 1990). Bu konuda araştırmacılar ikiye ayrılmıştır ancak Babbage Analitical Engine hakkında incelemelerini ve notlarını Lovelace ile incelemiş, kendisine üzerinde çalışma imkanı tanımıştır (Kim ve Toole, 1999). Lovelace, Analytical Engine için Bernoulli sayılarını hesaplayacak bir program geliştirme üzerine çalışmıştır. Lovelace ve Babbage Analytical Engine üzerine çalışmalarını mektupları ile sürdürmüştür ve bir mektupta Lovelage "Bernoulli numaraları hakkında motorla nasıl işlediğini gösteren notlar koymak isterim. Bana gerekli veri ve formülleri veriniz" der. Babbage'ın, Bernoulli sayıları programına ne ölçüde yardımcı olduğu bilinemese de bu Analytical Engine'e program eklenmesinin, Lovelage'ın fikri olduğunu göstermektedir (Kim ve Toole, 1999). 1840'da Babbage'ın Turin'de yaptığı sunumda, Luigi Federico Menabrea aldığı notlar ile "Sketch of The Analytical Engine" adıyla bir makale yayınlamıştır. Daha sonra Lovelace çalışmalarına bu makale üzerinden devam etmiş ve kendi notlarını ekleyerek makaleyi çevirmiştir (Kim ve Toole, 1999). B. Y. Bowder 1953'te yayınladığı Faster Than Thought kitabında bu makaleye yer vermiş, bu sayede Lovelace'ın çalışmaları ortaya çıkmıştır. Tüm bunlar düşünüldüğünde, bir makinenin programlanmasında Ada Lovelace'ın da önemli katkılarının olduğunu düşünmek yanlış olmamalıdır.

Lovelace makinenin insanların düşünme yöntemi ile düşünemeyeceğini ve herhangi bir şeyi ortaya çıkaramayacağını savunmaktadır. Makinenin, insanın verdiği düzen doğrultusunda çalışabileceğini dolayısı ile herhangi bir analitik ilişki veya gerçeği tahmin etme gücünün olmadığını kabul etmiştir (Ganascia, 2011). Lovelage'ın bir bilgisayar ve bir programın olmadığı dönemdeki düşünceleri onun programlama konusunda bugün hala geçerliliği olan yönde düşündüğünü göstermektedir.

20. yüzyılın başlarında bilgisayarın teorik temelleri akademi dünyasında atılmış, üniversiteler kendi cihazlarını geliştirmek için çalışmalara başlamıştır. 1930'da MIT'de Vannevar Bush, ilk modern analog bilgisayarı geliştirmiş, Harvard Üniversitesi profesörü Howard Aiken, Charles Babbage'ın Analytical Engine'ini incelemeleri üzerine farklı teknolojilerde dört hesap makinesi planları hazırlamış, 5 ton ağırlığında, 15 metre uzunluğunda, 750.000 parçalık ilk tam otomatik büyük ölçekli, hesap makinesini geliştirmiştir (Url-21). 1936 yılında ise Alan Turing yeni bir dönüm noktası olacak olan Turing Machine'i tanımlamıştır.

1930'lu yıllarda bazı problemler için algoritmik bir çözümün mümkün olmadığını ispat etme girişimleri ile bağlantılı olarak, algoritmanın matematiksel tanımlaması ve karakterize edilmesi ihtiyacı duyulmuştur (Davis, 2016). Bu amaçla birbirinden farklı formüller geliştirilse de Alan Turing'in soyut makine yaklaşımı tüm algoritmalar için ortak bir çözüm sunmakta olup modern çok amaçlı bilgisayarın gelişmesinde önemli bir adım olmuştur (Davis, 2016). Turing, araştırmasına matematiksel bir mantıkla, modern matematiğin önemli isimlerinden David Hilbert'in 1900 yılında Paris Uluslararası Matematikçiler kongresinde tanımladığı Entscheidungs problemi ile başlamıştır (Nuriyev ve Sadıgova, 2002). Hilbert'in yayınladığı 23 sorudan, onuncusu, M.S. 200 yıllarında yaşadığı tahmin edilen Diophantus'un birinci, ikinci ve üçüncü dereceden bilinmeyen denklemlerinin rasyonel tam sayılar ile çözümü olup olmadığı üzerinedir (Hilbert, 1990). Turing, Hilbert'in sorusunu, 1936'da On Computable Numbers, With An Application The Entscheidungproblem makalesinde, "hesaplanabilir sayılar" kavramı ile herhangi bir algoritmayı bir makinenin okuyabileceği "Universal Turing Machine" teorisinde açıklar ve Hilbert'in sorusunun çözümü olmadığını kanıtlar (Turing, 1936). Turing, hesaplanabilir sayıları, ondalık olarak ifadeleri sonlu, gerçek sayılar olarak tanımlamaktadır ve ancak bir makine tarafından ondalığı yazılabilen sayının hesaplanabileceğini düşünmektedir (Turing, 1936). Buna göre Hilbert denklemlerinde çıkan irrasyonel sayılar Turing'i, hesaplanamayan sayılar oldukları için, denklem çözümlerinin olmadığı sonucuna götürür ve ancak hesaplanabilir sayılarla tüm algoritmalara cevap verebilecek soyut bir makinenin tanımını yapar. Bir Turing makinesi, hücrelere bölünmüş sonsuz bir banttan ve bandı tarayabilen, okuma ve yazma işlemi yapabilen mekanik başlıktan oluşan bir cihazdır (Aspray, 1990). Bant data okuyucu veya hafıza olarak çalışmaktadır ve bant üzerinde iki yöne hareket edebilen başlık, hücredeki sembolü

okuma, okuduğu sembolü düzenleme ve hücredeki sembole bağlı olarak komşu hücreye geçiş yapma olmak üzere üç işlem yapabilmektedir (Url-24). Makine her algoritmanın kurallarına göre farklı işlem yapabilme özelliğine sahiptir. Ayrıca makinenin algoritmadan gelecek "dur" komutu ile işlemi sonlandırma özelliği de bir diğer önemli noktadır. Dur komutu olmadan işlem sonsuza kadar devam eder ve görev hiçbir zaman tamamlanamaz. Buna göre bir Turing Machine işlemi şu şekildedir: q_0 işlemin ilk ifadesi olup q_1 ve q_2 işlemde takip edilecek adımlar ve q_H durdurma ifadesidir. İfadeler arasında, tape üzerinde hareket eden başlığın hücrede okuduğu sembole göre devreye girecek komutlar yer alır. Komutlar 1/0, 1/1, 0/0, 0/1, 1/., 0/. ifadeleri ile belirtilip okunan *değer/yazılacak* değer anlamına gelmektedir. Başlığın hangi yöne devam edeceği komut sonuna yazılan R,L ile anlatılmaktadır (Url-23-24) (Şekil 3.7).



Şekil 3.7: Turing Machine işlemi, yazar tarafından yapılmıştır.

Evrensel Turing Makinesinin bilgisayar bilimleri için önemi, dijital, bilgi depolanabilen, bir dil ile talimatlar verilebilen, programlanabilen, bir bilgisayarın teorik bir modeli olması ve "otomatize teorisi"nin bir başlangıç adımı olmasıdır. Bu model ile bilgi saklanır, işlenir ve dijital olarak aktarılır (Aspray, 1990). Mantıksal

süreci işlemede, bu tarihten önce tek amaca yönelik geliştirilmiş tüm aletlerden, çok daha fazla gelişmiş bir makine tanımı yapılmıştır.

Alan Turing 1950'de yazdığı Computing Machinery and Intelligence makalesine "Makineler düşünebilir mi?" sorusu ile başlar ve bu sorunun cevabını "makine" ve "düşünme" kelimelerinin anlamları ile değil "imitation game" (taklit oyunu) ile arar (Turing, 1950, s.443). Bu test Turing'in, bilgisayar ve insanın aynı düşünme yetisine sahip olup olmadığını ölçme amacı ile sorular sorarak, cevapların bilgisayara veya insana ait olduğunu tahmin etme oyununa verdiği addır. Alan Turing'in "imitation game" ile açıklamaya çalıştığı nokta, bilgisayarlara düşünme yetisinin kazandırılabileceği olmuştur. Makalesinde, Ada Lovelage'ın bilgisayarın yapabildiklerinin, insanın onu geliştirebildiği kadarla sınırlı olduğu düşüncesinin, aslen "öğrenen makine" fikri ile sınırsızlaşabileceğini savunmuştur (Turing, 1950). Turing bugün hala tartışılan ve geliştirilmeye çalışılan yapay zeka araştırmalarının temelini oluşturmuştur.

Zaman içerisinde ilk bilgisayar fikri olan otomatik bir hesap makinesi için yanlızca sayılacak değerler büyümemiş, sayılacak bilgi de değişime uğramış ve saymak matematiksel işlem olma sınırından çıkmıştır. Charles Babbage'ın hesap makinesi icadı ile temelleri atılan, Alan Turing'in Turing Machine'i ile yapay bir dilin, programlamanın sınırlarını genişlettiği bilgisayarlar, bugün çok fonksiyonlu, pek çok bilgi ve işlemi depolayabilen, çok sayıda aritmetiksel işlemi kendisine yüklenmiş programa göre işleyebilen, sonuçlandırabilen cihazlardır. Jacquard loom dokuma makinesi ile keşfedilmiş makineler ile iletişim, makinelerin keşfine öncülük etmenin yanı sıra, Casey Reas'in ifadesi ile, bugünün bir dil ile programlama dünyasını oluşturmuştur ve bilgisayar artık bir araç olmanını ötesinde bir ortamdır (Reas, 2010).

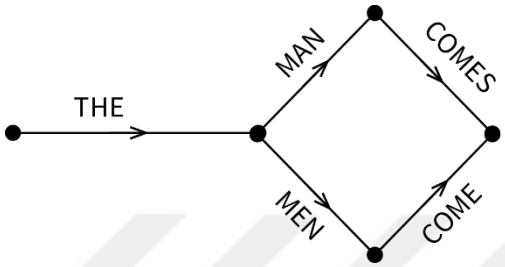
3.2 Kurallı, İşlemsel, Algoritmik Dil

Kayda geçmiş tarih itibari ile insan sayılarla, algoritma kullanarak çalışmıştır (Davis, 2016). Bunun sebebini anlamak algoritmanın kendi tanımı içerisinde mevcuttur. Martin Davis'in tanımına göre algoritmalar, hiçbir yaratıcı düşünceye ihtiyaç duymadan, başka bir ifade ile olağanüstü bir eylem olmayarak tamamen mekanik yöntemler ile bir talimat kümesini takip eden işlemlerdir (Davis, 2016). Başka bir ifade ile algoritmaların olağanlığı, sayıların kesinliği ile oluşmakta olup, işlemsel düşünmenin sonuç ürünüdür.

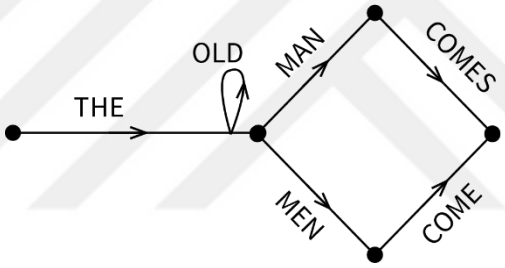
İşlemsel düşünce ve soyut düşünce, bilginin anlaşılması için gerekli akıldır. Bilginin çözümlenme ve işlenme süreci Marr teorisine göre bilgi işleme sistemi olarak adlandırılıp, "hesaplama teorisi" (computational theory), "algoritma" (algorithm) ve "uygulama" (implementation) olarak üçe ayrılır. Marr, bilgiyi soyutlamanın hesaplama evresinde olduğunu, eldeki verilerin ve çıktıların tanımlanıp, verilerin istenilen çıktılara nasıl dönüştürüleceğinin tanımının algoritma evresinde olacağını ve sonuç olarak algoritmanın fiziksele dönüşmesinin uygulama evresinde yapılacağını söyler (Marr, 1982). Jeannette M. Wing de, "Algoritma, bir sürecin istenilen sonuçları üretmesi için adım adım soyutlanmasıdır" olarak yaptığı tanım ile Marr'ı desteklemektedir (Wing, 2008, s.3718). Aslen tüm bu tanımların yanında basit bir ifade ile algoritmanın, "yöntem" olduğunu söyleyebiliriz.

Algoritmaları ifade etmek için bir metin kullanılır ve bu metinler bir dil ile yazılır. Her dilin olduğu gibi algoritma dilininde bir sözlüğü ve sözdizimi vardır (Coates, 2010). Paul Coates algoritmanın soyutlanarak gerçek bir dilin oluşması ile birlikte, dilbilgisinin üretkenliğinin algoritma ve sonuç arasındaki mesafeyi arttırdığını ve sonsuz çeşitliliğe erişim sağlandığını savunur. Üretken dilbilgisi tanımını Chomsky'nin yorumu ile yapmaktadır. Chomsky dilbilgisi tanımını "..bir dilin bütünü ve yanlızca doğru sözdizimi ile oluşturulmuş cümleleri" olarak yapar (Coates, 2010, s.2'de atıfta bulunulduğu gibi). Doğru sözdizimi (sentaks), Coates'in de vurguladığı gibi önemlidir. Çünkü doğru sözdizimi olmadığı takdirde dil hiçbir anlam ifade edemez (Coates, 2010). Chomsky'nin "finite state grammar" sonlu dilbilgisi adını verdiği dil içerisindeki sonlu sayıda kuralla etkileşim, sonsuz sayıda yapı üretmektedir ve Chomsky bunu bir makinenin kuracağı cümle örneği üzerinden anlatır: sonlu sayıda farklı durum arasında, bir sembol üreterek bir durumdan diğerine geçiş yapan bir makine olduğunu varsayalım. Makine bir başlangıç durumundan bir dizi durumu geçerek son hali aldığında belirli bir dili, başka bir ifade ile cümleler kümesini tanımlamaktadır. Üretilen dil ise sonlu bir dildir. Sadece iki cümle üreten makine şeması Şekil 3.8'deki gibidir (Şekil 3.8). Ancak kapalı döngüler eklenerek sonsuz sayıda cümle üretmek için dilbilgisi genişletilebilir (Şekil 3.9). Böylece "the old man comes" (yaşlı adam gelir) cümlesinin yanı sıra "the old old man comes" (yaşlı ihtiyar gelir), "the old men come" (yaşlı adamlar gelir), "the old old men come" (yaşlı ihtiyar adamlar gelir) alt cümleleri üretilebilmektedir (Chomsky, 1957, s.19). Coates bu tanımlamanın üzerine, belli bir sayıda kelime ve belli bir sayıda sözdizim kuralı

düşünürsek, bir dil içerisinde belli bir sayıda doğru cümle üretmemiz neredeyse imkansız olduğunu, matematiksel bir değeri olsa da bunun çok büyük bir sayıda olacağını söyler. Sonuçta çok az sayıda sözcük ve kuralı olan bir dil için bile üretilebilecek cümle sayısı çok büyük iken, algoritma temelli tasarım ve fikirler için çok daha büyük oranda üretim olasılığı oluşmaktadır (Coates, 2010). Bu sebepledir ki, algoritma ile kuralları belirlenmiş bir sürecin sonuç ürünlerinin sonsuz çeşitlilikte olması, bir dilinin olması sebebiyledir diyebiliriz.



Şekil 3.8: "finite state grammar" sonlu dilbilgisi şeması (Chomsky, 1957, s.19).



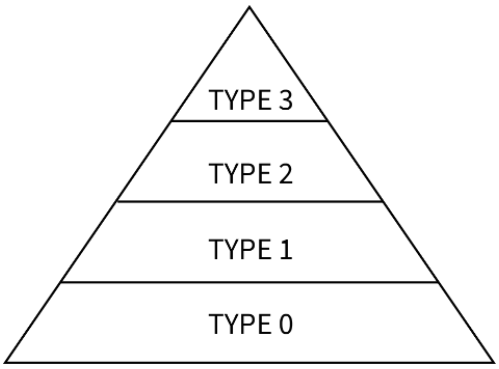
Şekil 3.9: "finite state grammar" sonlu dilbilgisi kapalı döngü eklentisi şeması (Chomsky, 1957, s.19).

Modern bilgisayarın ortaya çıkması ile insanlar, sayılar, isimler, resimler, ses dalgaları gibi her türlü bilginin dizi olarak temsil edilebileceğini fark etmişlerdir, ardından da diller olarak bilinen dizeler bilgisayar biliminin merkezi olmuştur (Jiang ve diğ, 2009). Kurallı, işlemsel, algoritmik, muntazam, biçimsel dil olarak kullandığımız, "formal language" teorisinin, Noam Chomsky'nin 1950'lerde, doğal dillerin yapısının kesin bir karakterizesini oluşturma girişimleri ile geliştiği kabul edilmektedir (Jiang ve diğ, 2009). Chomsky'nin amacı basit ve kesin matematiksel kurallar kullanarak dillerin sözdizimini tanımlamaktır. Daha sonra Chomsky'nin "context-free grammars" (bağımsız dilbilgisi) olarak adlandırılan dilbilgisi modeli kullanılarak programlama dillerinin sözdizimi tarif edilmiştir (Jiang ve diğ, 2009). Şadi Evren Şeker'in desteklediği üzere, kurallı diller, belirli harfleri olan, harflerin oluşturduğu kelimelerin oluşma kuralları ve kelimelerin dizilim kuralları tanımlanmış dillerdir (Url-25). Noam

Chomsky'nin tanımlamasına göre kurallı diller seviyelerine göre şu şekilde sınıflanmıştır (Url-25):

- *Type 0 unrestricted grammers (sınırlandırılmamış dil kuralları):* Dil L, dili oluşturan harfler a ile ifade edilecek olursa a* ifadesi a ile türetilen L anlamına gelmektedir. Tip 0 tüm dilleri kapsamakta olup oldukça esnektir. Herhangi bir dilde tüm algoritmaları işleyebilen Turing Machine ile işlenebilmektedir. Oluşan kelimelerin sonlu olma şartı yoktur, bu sebeple Turing Machine bandında sonsuz döngü içerisinde kelimeler üretilebilmektedir.
- *Type 1 context-sensitive grammers (içerik duyarlı dil kuralları):* Turing makinesi ile işlenebilen ancak bant uzunluğunun sabit olduğu kabul edilen dillerdir. Başka bir ifade ile üretilecek kelimelerin sabit zaman sonra sona ereceği kabul edilmektedir.
- *Type 2 context-free languages (içerikten bağımsız dil kuralları):* CFG olarak kullanılmakta olup tüm programlama dillerinin temelini oluşturmaktadır. S devamlılar a sonlular olarak ifade edilirse S→a S'nin a ile gösterileceği anlatılır. Bir başka ifade ile S=a dır. Diildeki tüm anlamlar belirlidir, çeşitli durumlarda belirsizlik içeren doğal diller için kullanılamaz.
- *Type 3 regular grammers (düzenli dil kuralları):* Tanımlı işlemlerdir. Sonlu alfabe içinde, sabit kural ve değerler mevcuttur.

Seviyeler arasında ileri gidildikçe, özelliklerinden de anlaşılacağı üzere kurallar artmakta ve esneklik azalmaktadır. İfadelerin karşılıkları netleşmeye başlar. Aynı zamanda üst seviyedeki diller alt seviyeyi kapsamaktadır (Şekil 3.10). Buradan çıkarılacak sonuç ise, bir dil içinde kuralların artmasıyla, bir ifadenin o dil içerisindeki karşılıklarının netleşmeye başladığıdır. Bu da doğal dillerden çok daha tanımlı ve kurallı olan yapay dillere doğru bir akışı ifade etmektedir. En üst seviyede kabul edebileceğimiz doğal diller tüm yapay, programlama dillerini kapsarken, kelimelerin birden çok anlamı içerme durumu yapay dillerde tamamen ortadan kalkmakta olup, her ifadenin, kesin karşılığı oluşmaktadır.



Şekil 3.10: Chomsky hiyerarşisi.

3.2.1 Algoritmik düşünce

Bir problemin verilerinin oluşturulduğu hesaplama için işlemsel düşünce, bu problemin sonucunun çözümlendiği algoritma için algoritmik düşünce gereklidir. Algoritmik düşünce, hesaplamanın mantığını çözümlememiz için gerekli akıl olan işlemsel düşünce gibi algoritmayı anlamak için gerekli düşüncedir. Bu sebepledir ki, işlem yapmanın da dışında bir düşünüş olan işlemsel düşünce gibi algoritmik düşünce için algoritma ve programlamadan bağımsız olarak bir düşünce, bir yaklaşım yöntemidir demek yanlış olmamalıdır. Gerald Futschek kısa bir ifade ile problem çözmek diyebileceğimiz algortima için gerekli akıl; algoritmik düşüncenin algoritmayı oluşturmak ve anlamak için gerekli yetenek havuzu olduğunu ileri sürerken, verilen problemi analiz etme, problemi tanımlama, verilmiş probleme uygun temel fiilleri bulma, doğru algoritmayı oluşturma, problem için tüm durumları düşünme ve algoritmayı geliştirmenin bu havuzda mevcut olması gereken yetenekler olduğunu düşünür. Tüm bunların sonucu olarak da aslen algoritmik düşüncenin verilen problemi çözmek için yeni algoritmalar oluşturabilme durumu, başka bir ifade ile türetici bir yetenek olduğunu ileri sürer (Futschek, 2006). Casey Reas algoritmayı şu dört özellik ile tanımlamaktadır (Reas, 2010):

- Bir algoritma yazmak için birden çok yol vardır.
- Algoritma varsayımlar gerektirir.
- Algoritma karar gerektirir.
- Kompleks algoritmalar parçalara bölünmelidir.

Bu özellikler ile birlikte algoritmaların hem objektif hem de subjektif işlemler olduğunu anlamaya başlıyoruz. Aslen problem çözümünde gerekli karar alma, varsayımda bulunma algoritmanın öznel adımlar ile şekillenmesi ve bu durumda bir

problem için pek çok çözüm yöntemi geliştirilebileceği, çözümün tanımlı, kurallı bir dil ile, programlama dili ile ifadesi sayesinde de soyutlamanın öznel bir işlemi genel duruma getirdiğini görmekteyiz. Bu durumu bir örnek üzerinden açıklamak mümkün: problemin bilgisayarın çalışmaması olduğunu kabul edelim. Bu durumda A kişisinin problem için sebep ve çözümleri şunlar olabilir:

1 Bilgisayarın şarjı bitmiştir, şarj etmelidir.

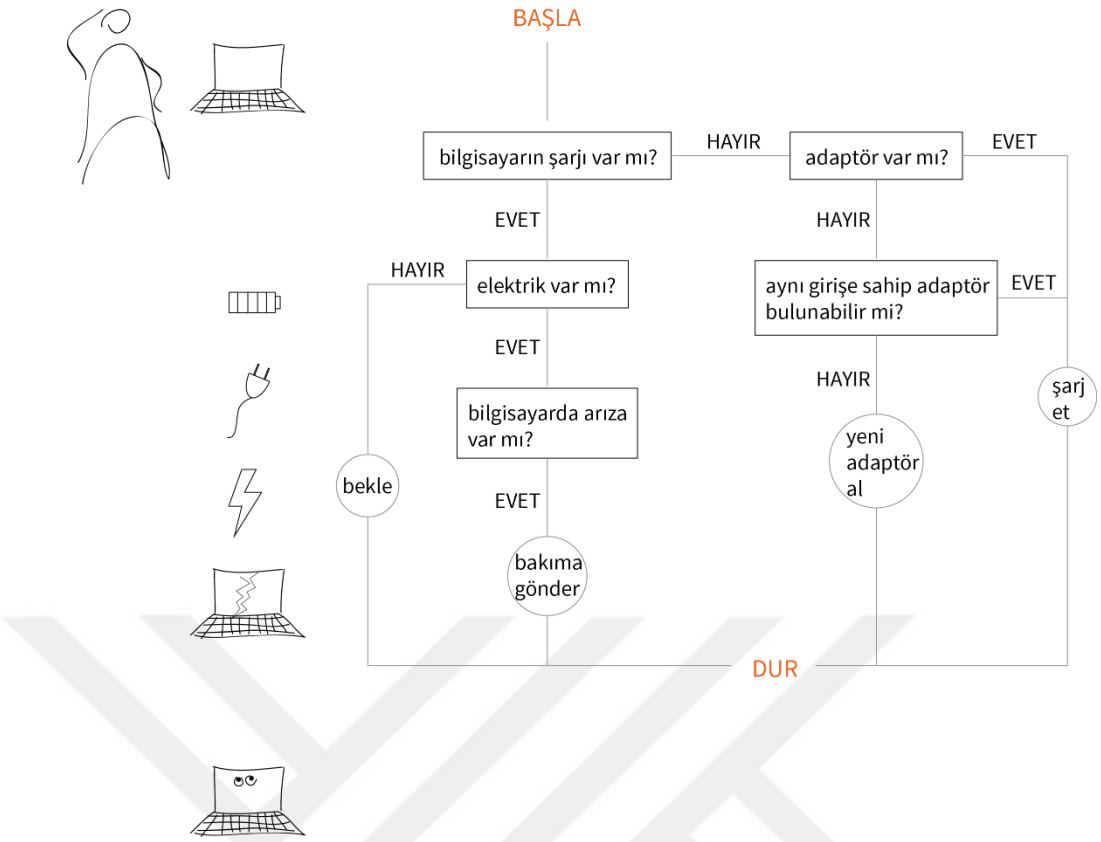
2 Bilgisayarın adaptörü yanında değildir, aynı girişe sahip başka bir adaptör ödünç almalıdır.

B kişisi için:

3 Elektrikler kesilmiştir, elektriklerin gelmesini beklemelidir.

4 Bilgisayarda bir arıza vardır, bakım yaptırmalıdır.

Bir algoritma tüm olasılık ve çözümleri kapsamalıdır ve problem için varsayımlarda bulunarak algoritma yazılır. Bu durumda çözüm şeması Şekil 3.11'deki gibidir (Şekil 3.11). Fabian Scheurer ve Hanno Stehling'in tasarım problemi üzerinden algoritma tanımı yapar. Bu örnekte olduğu gibi öncelikli olarak problem iyi tanımlanmalı, eğer takip edilmesi gereken adımların bilgisayarlara verilen komutlar olduğu düşünülürse, deneyimlere ve iç güdülere göre hareket etmeyen, dolayısı ile tahminde bulunamayan bilgisayarlar için, her adımın bir önceki adımda tam olarak ve belirsizlik olmayacak şekilde tanımlanması gerekmektedir (Scheurer ve Stehling, 2011). Bu tanımlama ile birlikte A ve B kişisine göre değişen çözüm önerileri olan bir durumdan, genel bir problem çözümü adımları belirlenmiş olur ve kişilerin subjektif durumları genelleşmiş olur diyebiliriz.



Şekil 3.11: Yazar tarafından verilmiş örnek problemin çözüm şeması.

Görülüyor ki, algoritmik düşünce yanlızca bilgisayarlar ile iletişim kurabilmek için sahip olunması gereken düşünce şekli olmanın ötesinde aslen çözümleme ve yöntem geliştirilmesi gereken her durumda devreye giren bir yaklaşımdır demek yanlış olmamalıdır.

Algoritmik düşüncenin anlaşılmasını algoritmik düşüncenin nasıl öğretildiği üzerinden yorumlamak mümkündür. Gerald Futschek bunun yaratıcılığı öğretmek kadar zor olduğunu ancak pek çok problem çözme yöntemi ile geliştirilebileceğini düşünmektedir (Futschek, 2006). Jeongwon Choi, Youngjun Lee ve Eunkyoung Lee de, problem çözümündeki soyutlamanın problem tanımı, farklı problem çözümleri ve en uygun sürecin algoritma tasarımını içeren kompleks bir yapı olduğunu, algoritmik düşüncenin algoritma tasarımı öğrenmek ile değil gerçek problemler deneyimlemeye bağlı olduğunu ve algoritma tasarımı öğrenmenin etkili bir problem çözümü geliştirilebileceğini garanti etmediğini, bunun ancak gerçek problemlere uygulanması ile gelişebileceğini savunmaktadır (Choi ve diğ, 2017). Gerald Futschek hiçbir programlama diline bağlı olmayan labirent problemi örneklemesi yapar (Futschek,

2006). Labirent içinde yol bulma görevinde basitçe anlaşılabilecek, ancak çözümün belirsiz olduğu üç durum belirtir:

- Labirentin dışında bir yol bul
- Labirentin içinde bir yol bul
- Labirentin içinde spesifik bir pozisyon için bir yol bul

Sonra bu görev için genel bir ifade yazar:

- A noktasından B noktasına bir yol bul

B noktasına ulaşırken labirent hakkında bildiklerimiz önemlidir. Basit bir senaryoda Futschek labirent için büyüklük, strüktür gibi hiçbir bilgimizin olmadığı ve bir sonraki kesişme noktasına gelene kadar koridor boyunca yürüdüğümüzü ve kesişme noktasına geldiğimizde bu noktaya bağlı tüm koridorları görüp önceden geçilmiş kesişme noktalarına benzeyip benzemediğini düşünerek karar vermemizin mümkün olmadığı kabul eder. Buna göre hangi koridordan gidilebileceğine basit bir strateji ile karar vermeye çalışır:

- En soldaki koridoru takip et
- Rastgele bir koridoru takip et
- En soldan başlayarak sistematik olarak tüm koridorları takip et

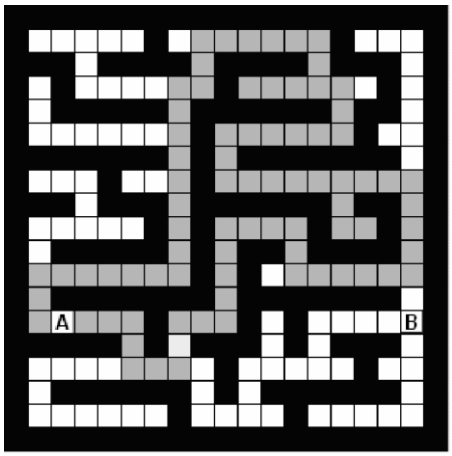
Birinci stratejide her zaman aynı yönünseçilmesi dairesel düzen oluşturmakta ve her kesişim noktasında doğal bir kural takibi yapılmaktadır. Bir kural dayanması sebebiyle yanlış koridorlar seçilip, labirent içerisinde bir koridor birden çok kez geçilebilse de Futschek bu stratejinin çalıştığını gözlemlemiş ve birinci stratejiyi 4 adımda tanımlamıştır (Şekil 3.12):

while (müddetçe) hedefe ulaşmadığın müddetçe

if (eğer) koridor kesişim olmadan bitti ise

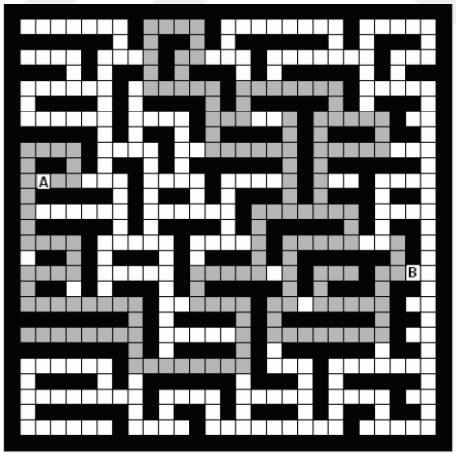
then (bu durumda) 180 derece etrafında dön **and** (ve) bir önceki kesişime git

else (değilse) sol kenarı takip et



Şekil 3.12: Sol kenarı takip etme algoritması (Futschek, 2006, s.164).

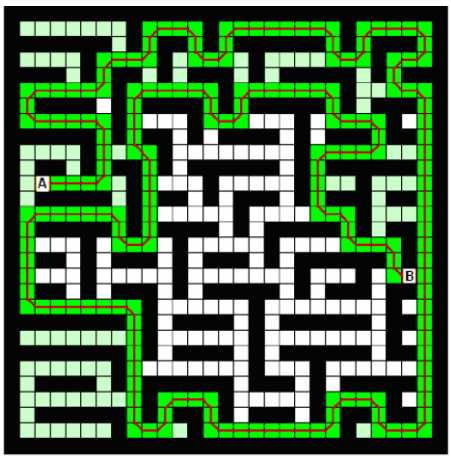
Birinci strateji ile B'ye ulaşılsa da, algoritma dairesel hareket doğası gereği sonsuza kadar devam edip bir döngüye girme yapısında. Döngüden kaçınmak için denenen ikinci strateji, rastgelelik içerip belirleyici bir yöntem olmasa da, her kesişimde baştan başlayan ve bir seçim gerektiren bir yöntem (Şekil 3.13). Ancak bu yöntemin her labirent için sonuç verip vermemesi bir kurala tabi olmadığı için belirlenememektedir. Bu noktada algoritmanın nasıl geliştirilebileceği, en kısa yolun nasıl bulunabileceği soruları devamlılığını sürdürmektedir.



Şekil 3.13: Rastgele seçime dayalı yol takibi algoritması (Futschek, 2006, s.165).

Son olarak Theseus'un Knossos'daki labirentte, Minotaur'u bulmak ve öldürmek için kullandığı ünlü "Ariadne's Thread" (Ariadne'nin İpliği) örneği verilmiştir. Bu bilinen en eski geri izleme algoritması olarak geçmektedir. İplik, başlangıç konumundan hedef konuma kadar olan yolu gösterir. Temel olarak yeni koridorda iken ipliğin çözülmesi, başarısız bir yol tekrar edilirken ipin tekrar bağlanması gerekir. Theseus labirent içinde gezerken kendi bağladığı ipliği bulduğunda bir döngüde olduğuna emin olur. Bu

sayede geri bildirim olarak adlandırılmış, bi başka ifade ile, önceki geçişlerin iplik yardımı ile kontrolü, döngüye engel olan bir algoritma oluşturmuştur (Şekil 3.14).



Şekil 3.14: "Ariadne's Thread" (Futschek, 2006, s.166).

Sonuç olarak Futschek, herhangi programlama dilinden bağımsız olarak farklı problemler ve problem üzerinde farklı algoritmalar geliştirerek, algoritmaların çalışma mantığı ve beraberinde algoritmik düşüncenin anlaşılabileceğini savunmuştur (Futschek, 2006). Futschek' in algoritmik düşünce için "..yaratıcılığı öğretmek kadar zor" ifadesi aslen bir düşünce sistemi olması sebebiyledir. Paul Coates "Eğer mimarlar sistem tasarımcıları iseler, fikirlerini geliştirmek için bilgisayarın çıktılarını gözlemlemek yerine, kendi algoritmalarını önermek için algoritmik düşünmelidirler" sözleri ile tasarım ile iç içe olan algoritmik düşünce sisteminin bir kez daha altını çizmektedir (Coates, 2010).

3.3 Programlama Dili

"...programlama kesinlikle teknik bir görev değildir; bir iletişim eylemi ve evreni temsil etmenin sembolik bir yoludur" (Reas, 2010, s.17).

Programlama, birçok tarihçi tarafından ilk program olarak kabul edilmiş; 1800'lerde Charles Babbage'ın Analytical Engine makinesi ile Bernoulli sayılarını hesaplamak için Ada Lovelage'ın geliştirdiği yöntem ile başlamıştır. Günümüze kadar Kondrad Zuse'nin Plankalkül, Alan Turing, John von Neumann gibi isimlerin ENIAC, John W. Backus'un FORTRAN, John McCarthy'nin LISP, Grace Hopper'ın COBOL, John George Kemeny ve Thomas EugeneKurtz'un BASIC, Niklaus Wirth'in PASCAL, Dennis Ritchie'nin C programlama dilleri, Microsoft'un Visual Basic, James

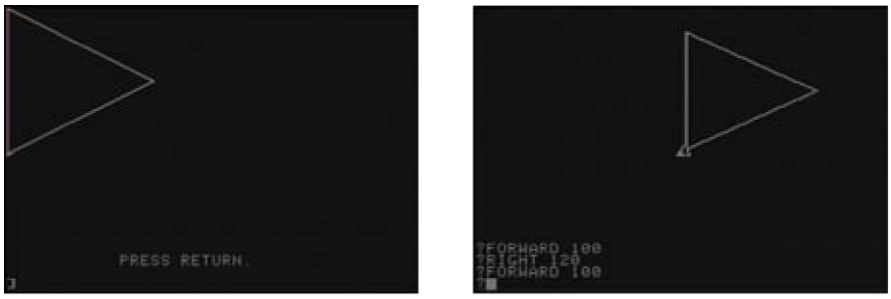
Gosling'in Java, web tabanlı ASP, JSP, PHP dilleri gibi pek çok kişi tarafından geliştirilmiş programlama dilleri ile bugün yapay zeka programları üzerine çalışılır seviyeye gelinmiştir (Akcay, 2013).

Programlama, oluşturulmuş bir algoritmanın, bir programlama dilinde formalleştirilmesi (kodlanması) ve bir bilgisayar tarafından algoritmanın yürütülme sürecidir (Url-26). Programlar, insan iletişim sürecine benzer şekilde, doğru algoritmalar yoluyla bir makinenin davranışını ve çıktısını kontrol etmek için programlama dilleri aracılığıyla oluşturulur (Url-27). Programlar ile iletişim kaynağı programlama dillerinin ise kendi mantıksal kurguları mevcuttur. Dolayısı ile birebir aynı algoritmanın her programlama dilindeki karşılığı farklı olmasının yanında, algoritmanın oluşturulması da farklılık gösterebilmektedir. Casey Reas bu farklılığı marangoz örneği ile dile getirmektedir:

> Her programlama dili, çalışmak ve düşünmek için farklı bir materyal türüdür. Bir marangoz meşe, balsa ve çam gibi malzeme çeşitlerinin, ormanda eşsiz özelliklerini bildiği gibi, yazılım konusunda bilgili bir kişi farklı programlama dillerinin benzersiz yönlerini bilir. Masayı inşa eden bir marangoz, masraf, dayanıklılık ve estetik gibi faktörlere dayalı olarak odunu seçecektir. Bir programcı tahmini bütçeye, işletim sistemine ve estetiğe dayalı bir programlama dili seçer. Her programlama dilinde sözdizimi (veya dilbilgisi), o dilde mümkün olanı yapılandırır. Farklı programlama dilleri, programcıların çalışmalarıyla ilgili düşüncelerini (veya eylem olasılıkları) bu dilin kısıtlamaları yoluyla teşvik eder. (Reas, 2010, s.17).

Farklı düşünce yapılarına sahip programlama dilleri, aslen insanın düşünce yapısında da etkiye sahiptir (Şekil 3.15). Reas, bir programlama dilinin belirli bir düşünme biçimi oluşturduğunu savunmakta olup programlama dillerinin akıl farklılıklarını BASIC ve LOGO dillerini karşılaştırarak anlatmaktadır:

> Bir programlama dilinin belirli bir düşünce tarzını teşvik etme biçimi, iki çok farklı dili karşılaştırarak gösterilebilir: BASIC ve LOGO. Bu programlama ortamlarının her birinde üçgen çizmek, farklı bir yaklaşım ve alan anlayışı gerektirir. BASIC, kurulu bir koordinat sistemine dayanır ve koordinat bilgisi gerektirir; çizgiler, bir koordinatı diğerine bağlayarak çizilir. Buna karşın, LOGO henüz geometri öğrenmemiş küçük çocuklar için geliştirilmiş bir dildir; Kullanıcının, şekilleri sadece açıları ve sol ile sağ arasındaki farkı anlayarak kodlamasına izin verir. Bu programda, çocuk bir kaplumbağanın yolunu ekranda yönlendirerek çizgiler çizer. Çocuğun, kaplumbağa olduğunu ve ileri doğru ilerlediğini, sağa döndüğünü, ileri doğru ilerlediğini, sağa tekrar döndüğünü ve üçgenin tamamlanması için ilerlediğini hayal eder. Her iki dil aynı şekiller çizmesine izin verdiği halde, LOGO araştırmayı desteklerken BASIC objektifliği geliştirir. Buna ek olarak, LOGO, programcıyı kodun zihinsel olarak çalıştırılmasını teşvik eder. (Reas, 2010, s.17-18).



BASIC
```
10 HGR : HCOLOR = 3
20 HPLOT 0, 0 TO 100, 50 TO 0, 100 TO 0, 0
30 END
```

LOGO
```
FORWARD 100
RIGHT 120
FORWARD 100
RIGHT 120
FORWARD 100
```

Şekil 3.15: LOGO-BASIC programlama dillerinde üçgen çizimi (Reas, 2010).

Paul Coates, dillerdeki düşünce yapısının bir adım geriye giderek dilbilgisi ile ilişkilendirmesini yapar. Bu fikrini kuvvetlendirmek için Alan Kay'ın dil kategorisinden destek alır. Alan Kay, Seymour Papert gibi etkileşim ile kendi kendine öğrenebilen bir makine tasarımı üzerine uzun yıllar çalışmıştır. Bu program için bir dil tasarımının filozofik bir eğitim gerektirdiğini savunmuştur. Programlama dillerinin kategorizasyonu için Kay, Papert'in yaklaşımını kullanmaktadır. Papert, programlama dillerini; zorunlu (imperative), uygulamalı (applicative), mantık temelli (logic-based), problem odaklı (problem-oriented) olmak üzere kategorize etmektedir, ancak dillerin ya özelliklerin birleşimi ya da stil kristalizasyonu şeklinde görüldüğünü düşünür. Özelliklerin eklemlendiği diller gruplar tarafından zaman içerisinde oluşturulurken, kristalize diller tek bir kişinin çalışmaları ile gelişmektedir. Fakat bu durumu Papert bilgisayar dillerinde dilsel olarak ayırmaktadır. McCarthy dilin sözdizimi (sentaks) kurallarının tanımlanmasından önce uygulama ayrıntılarının detaylandırılması, değişkenlerin değerlendirilen işlev içinde döndürülme şekli ve belleğin düzenlenmesinin tanımlanması gerektiğini açıkça belirtmektedir (Coates, 2010). Başka bir ifade ile farklı düşünce yapılarına sahip dillerin tanımında temiz ve tutarlı sözdizimi kurallarının net bir şekilde tanımlanmış olması gerekmektedir. Belirlenen kurallar doğrultusunda düşünce yapısının da geliştiği düşünülebilmektedir.

"Mimarlık dilinin" nasıl bir programlama dili olarak tanımlanabileceğine bakmak ilginç bir konudur. Bu konudaki dezavantaj, Paul Coates 'in de desteklediği üzere, programlama dillerinin tamamen açık ve mekanik aygıtlar olması ve çoğu tasarımcının, mimari yaklaşımlarını böylesine resmi bir biçimde kodlamaktan kaçınmasıdır. Bununla birlikte avantaj, bir kere tanımlandıktan sonra çok basit dillerin bile deney yapmamıza ve form üretmemize izin verebilir olmasıdır. "Çünkü bilgisayar

bir kez kendi dilini konuştuğunda yorulmadan dilde cümleler üretir" (Coates, 2010, s.121).

Casey Reas da konuya programlama dillerinin disiplinler arası durumu yönünden bakmaktadır. Reas'ın sözünü ettiği üzere, sanatta kullanılan birçok dilin başlangıçta bu alanlar için tasarlanmadığı talihsiz bir gerçektir. Tasarımcıların, mimarların ve sanatçıların yazılım arzuları aslen hem düşünüş hem uygulama farklılıkları sebebi ile çoğu zaman bilim adamları, matematikçiler ve mühendislerden farklıdır. Bu sebeple tasarımcıların, C ++ ve Java gibi en baskın dillerle görsel form oluşturmak için gereken teknik beceriyi, genellikle kazanmak yıllar alır (Reas, 2010). Programlama dillerininin geliştirildiği disiplinler arasında dahi tasarımcıların yeri oldukça azdır. Casey Reas ve Ben Fry'ın geliştirdikleri Processing görsel programlama dili, tasarımcıları programlama dünyasına yaklaştıran önemli programların başında olduğunu söylemek yanlış olmayacaktır. Reas'a göre görsel programlama dilleri (grafiksel programlama dili olarak da adlandırılır) kodla düşünmenin alternatif bir yolunu sunar. Bir programın görsel bir programlama dili ile yazılması, bir metin yazmak yerine diyagram yapmaya benzer (Reas, 2010). Bunu tasarımcıların metin temelli programlardan çok görsel programlama dillerine yönelme sebeplerinden biri olarak görebiliriz. Bu noktada teknik olarak tasarımcıların çoğunun kullandığı kodlama tipi de farklılık göstermektedir. Kod bilgisayarın donanımında algoritmayı yürüten programların metni için genel ifade olmakla birlikte, belirli bir kodlama türü olan betik, mevcut program içinde programı istenilen yönde dönüştürme odaklıdır. Daha basit bir ifade ile betik ile program yazılmamakta, ancak mevcut programlama dili kullanılarak program içinde algoritmalar geliştirilebilmektedir. İşlemsel tasarımda aslen mimarlar çoğunlukla betik yazmakta olup yazılıma müdahale çok daha basittir. Programın kendisi, tasarımcı için asıl amaç olmayan programlama için gerekli efor oranını, arttırılmış görselleme ile azaltmakta olup, kullanımı tasarımcının ana odak noktasına taşımaktadır. Bu sayede tasarımcı için programlama ile arasındaki mesafe azalmaktadır.

3.4 Kodlama

"Kodu geniş bir anlamda kurallı dil (formal language) olarak düşünürsek, gerçekliğin bir kod tarafından yönetildiğini söyleyebiliriz" (Borgmann, 2011, s.184).

İşlemsel düşünce başlığı altında, soyut düşünce, kurallı, işlemsel, algoritmik dil ve programlama dilinden sonra son adım olarak düşünülebilecek kodlama, programlama dilleri kullanılarak oluşturulmuş kural dizeleridir. Başka bir ifade ile tüm kavramların bir arada tanımını yapmak gerekirse; "soyut mantık"la oluşturulan problem çözüm yöntemi "algoritma"nın, makinaya kendi dili olan "programlama dili"nde anlatımının yapıldığı metin dizesi "kod"dur denilebilir. Kod için bilgisayar dilindeki metin tanımlaması, Paul Coates için, aslen "bir cümle içine sığdırılmış açıklama" benzetimidir (Coates, 2010). Bu benzetime göre kodun 3 satırlık özeti şu şekildedir:

```
to repel
do something
end
```

Kod içerisinde bilgisayara bir şeylerin nasıl yapılacağı tanımlandığı için bilgisayarın tanımlanan şeyi nasıl yürüteceği ortaya çıkarılmalıdır (Coates, 2010, s.7). Bunu daha iyi açıklamak için Coates'in turtle kodu örneği verilebilir. Turtle kodunda çok sayıda nokta rastgele 2 boyutlu bir yüzey üzerindedir ve her noktaya bir görev tayin edilir:

Diğer tüm noktalara bak ve kendine en yakın olan noktayı bul.

Sonra bulduğun bu noktadan uzaklaş.

Bu işlem esnasında noktalar en yakın komşularından uzaklaşırken yanlışlıkla başka bir noktaya çok yaklaşabilir, ancak işlemin devamında yeniden noktalara bakıldığında en yakın nokta olarak yaklaşılmış noktalar bulunacak ve bu kez de bu noktalardan uzaklaşılacaktır. Coates bu algoritmayı NetLogo dilini kullanarak bilgisayara anlatır. Noktalar "kaplumbağalar" kullanılarak tanımlanır ve aslında hepsi ayrı birer bilgisayar programıdırlar. Kod şu şekildedir:

```
to repel
ask turtles
{
set closest-turtle min-one-of other turtles [distance myself]
set heading towards closest-turtle
back 1
}
```

end

"to" ve "end" sözcükleri arasında uzaydaki çok küçük soyut bilgisayarlar olan gözlemci kaplumbağalara üç cümle ile mesaj gönderilmektedir ve kaplumbağalara şöyle seslenilmektedir:

Sevgili kaplumbağalar,

Mesafesi en az olan kaplumbağaları bulmak için diğer tüm kaplumbağalara bakmanızı rica ederim.

Kaplumbağaların kendilerine yakın olanları hatırlamaları için bilgiyi kayıt edebilecekleri, geri dönebilecekleri bir referans gerekmektedir. "closest-turtle" referansı betik içerisinde tanımlanarak aslen bilgisayar ile ortak bir alan oluşturulur (Coates, 2010, s.7).

Turtle kodu ile Coates aslen üçgenleme deseni çıkarmaktadır. Kodun içeriği üçgen bilgisinden uzak, yazılan programlama dilinin düşünce sistemine uygun, soyut bir hikayeye dönüşmektedir. Bu noktada Coates'in "distributed representation" (dağıtılmış temsil) olarak adlandırdığı bir durum oluşmaktadır. Kaplumbağalar üçgen deseni çıkardıklarının farkında değildirler. Onlar kimin kendine yakın oldukları ile ilgilenirken, monitöre bakan göz, desen ile ilgilenmektedir. Başka bir ifade ile, farklı seviyelerde gözlem ve gözlemciler mevcuttur (Coates, 2010, s.43). Bu da aynı kod üzerinde, makine ve insanın farklı düşünce yapısı ile ancak ortak metni takip ederek çalıştığını göstermektedir. Makine ve insandan biri diğerinin yerine geçmezken, aralarında bir işbirliği söz konusudur.

Turtle kodundaki "closest-turtle" tanımına geri dönülecek olursa, bu ifade betiğin bir değişkenidir. Algoritma çalıştığı müddetçe, kaplumbağalar her yer değişimi sonrasında yeni bir ölçüm daha yapar ve kendilerine en yakın kaplumbağalar bir önceki ölçüm sonucu yapılan hareket dolayısı ile değişmiştir. Dolayısı ile "closest-turtle" sabit bir değer değil, ilişkilere göre sürekli değişen bir değerdir. Bir kod içerisinde değişkenlerin tanımı şu şekilde yapılabilir (Jabi, 2013, s.23):

- integer: Tam sayı değişkenlerdir (1, 2, 7,...) Genellikle kod içerisinde "sayma" için kullanılabilir.
- float: Ondalıklı değişken değerleridir (1.1, 12.5, ...). Ölçüm için kullanımı idealdir.

- string: Karakterleri ifade eder (kelime, cümle gibi) ve kaydetme, işleme, bilgiyi gösterme işlemleri için kullanılır.
- array: Pek çok değişken ve öğeyi kaydedebilen konteyner gibidir (merkezleri x_1, y_1, z_1 koordinatında r_1, r_2, r_3, r_4 olan 4 adet daire). Genellikle listeler ifade edilir.
- boolean: true-false , 0-1 gibi ikili değeri olabilen değişkenler için kullanılır.

Değişken değer aslen işleyişin ve sonucun ana kaynağıdır. Casey Reas değişkeni "..bir işlemin çıktısını etkileyen bir değer", "değeri değişebilen" olarak tanımlamaktadır (Reas, 2010). Sonuç çıktısı üzerinde direkt etkisi olan değişkenin, kod içerisinde tanımı ve kontrolünün, kodlamanın en önemli noktalarından biri olduğunu söylemek yanlış olmamalıdır. Temelde tüm işlemin, üzerinden ilerlediği değişkenlerin, kodlamada sabit değerlere sahip olmayan esnek yapısı, aslen bugün tek bir forma, tek bir problem çözümüne gebe olmayan tasarım ve diğer tüm disiplinler için neden kodlama sorusunun cevabı ve kodlamanın temel mantığıdır. Bir şeyi değişkenlerle ifade etme, o şeyi parametrelerle ifade etmektir (Reas, 2010). Başka bir ifade ile tanımlanacak her yeni duruma uyumlu yapıya sokmaktır. Casey Reas simülasyon algoritması üzerinden ise şu tanımlamayı yapar; "Algoritmada üç bölüm vardır: değişkenler, bir sistem ve bir durum. Değişken bileşenleri tanımlayan değerdir, sistem değişkenlerin nasıl etkileşimde olacağını tanımlar ve durum sistemin verilen zamanda değişkenlerindeki değerdir" (Reas, 2010, s.13). Aslen bu tanımlama da algoritmanın tamamında değişken üzerine kurulu bir sistem anlatmaktadır. Değişken sistemi ile kodlama sonuç ürünündeki değer aralığını arttırmakla birlikte geniş bir olasılık aralığı sunmaktadır.

3.4.1 Tasarımda kodlama

> Sonucun büsbütün kontrolünde olmak isteyen tasarımcı, sürecin kontrolünde olmalıdır. Sürecin kontrolünde olmak için tasarımcı, araçların kontrolünde olmalıdır. Araçlar hesaplamalıdır, bu yüzden kontrolü ele almak isteyen bir tasarımcı da bir kod yazarı (scripter) olmalıdır (veya diğer insanların araçlarını kullanmanın görünmeyen etkisinin sonucundan muzdarip olmalıdır). Robert Aish (Burry, 2011, s.67'de atıfta bulunulduğu gibi).

"Mimarlık kodlanır ve herzaman da kodlanmıştır" (Rocker, 2006, s.21).

Tasarlama, insanın tasarlanmış doğadan öğrendiği bir fiildir. İnsanın tüm etkilerine karşı sürekli bir tepki ve değişim hali içindeki doğa, hiçbir zaman sabit bir formülün

ifadesi değildir. Doğadan öğrenilmiş tasarlama ile doğaya verilen tasarımların da sabitliği aslen evrenin işleyişine taban tabana zıt bir duruştur. Daniel Bosia, tasarlamanın derinlemesine bir form yapılanmasının, doğal yapılar gibi doğrusal olmayan bir süreç olduğunu savunur ve Bosia'ya göre doğrusal olmayanlık, tek cevapları değil birden çok sonucu üretir (Bosia, 2011). Bu yaklaşım, değişken yapısı ile sonsuz sonuç üretme potansiyeline sahip kodlamanın doğasında mevcut bir şekilde bulunmaktadır. Aslen tasarım ve programlamanın buluştuğu payda da bu yaklaşımdır demek yanlış bir ifade olmayacaktır. Ancak bundan bir adım geriye gidilecek olur ise, buluştukları daha genel bir payda vardır; problem çözmek. Descartes'e göre problem çözme tekniği şu şekildedir (Ayten, 2010, s.68):

1 Doğruluğu kanıtlanmadıkça birşeyi doğru kabul etmeyin; tahminden ve önyargıdan kaçının.

2 Problemi mümkün olduğu kadar çok parçaya bölün.

3 Düzenli bir şekilde düşünün; anlaşılması kolay noktadan başlayıp daha karmaşık olanlara doğru ilerleyin.

4 Olaya bakış açınız oldukça genel, ancak hazırladıklarınız, ayrıntılı ve hiçbir şeyi dışarıda bırakmayacak kadar eksiksiz olsun.

Dr. Umut Engin Ayten, Descartes'in problem çözümü adımları ile programlama sürecindeki problem çözümünü kıyaslayıp Descartes'in tekniği ile benzerliğini göstererek problem çözme sırası oluşturmuştur (Ayten, 2010, s.68):

1 Problemi anlama (understanding, analyzing)

2 Çözüm yolu geliştirme (designing)

3 Algoritma ve program yazma (writing)

4 Tekrar tekrar test etme (reviewing)

Ayten'in programlama için tanımladığı adımlar, tasarım sürecinde de geçerlidir. Yüzlerce yıllık geçmişi olan tasarım süreci ancak 1960'larda birbiri ile iç içe olan 4 adım ile formüle edilmiştir (Kalay, 2004, s.7):

- Problem analizi (problem analysis)
- Çözüm sentezi (solution synthesis)
- Değelendirme (evaluation)

- İletişim (communication)

Ancak işlemsel tasarım, iki alanın düşünce ve yaklaşım benzerliğini, dillerine de yansıtma girişimindedir. “tasarım dili” “programlama dili” ile buluşur. Paul Coates tasarım sisteminin oluşturulması için önerilen modelin, tasarımın yapı ve anlamını ifade eden ve tartışılabilir kılan kurallı bir dilde (algoritma ile) tanımlanması gerektiğini, modellemede kurallı dillerin bu şekilde kullanılmasının, geleneksel yöntemlerden daha fazla kabul gören ve “tartışılan” bir biçim ve şekil teorisinin gelişmesi için temel oluşturduğunu savunmaktadır (Burry, 2011). Çünkü formun oluşmasının altında yatan tüm varsayımlar açıkça ve gizlenmeksizin kurallara tabi tutularak ortaya konulmaktadır.

Algoritmik mimarlık olarak çevirebileceğimiz algorithmic architecture, programlama ve tasarım dilinin bütünleştiği bir alan olarak aslen oldukça geç oluşmaya başlamıştır. Wolfram’ın hesaplama alanındaki araştırmaları, Alan Turing’in Turing Machine (1936) ve Aristid Lindenmayer’in L-sistemleri (1968) gibi daha önceki modellere dayanmakta olup, bu modeller algoritmik mimari ile çağdaş tartışmalarda gittikçe artan oranda önem kazanmaya başlamıştır ve 1960’lar alanın yavaş yavaş oluşmaya başladığı dönem olmuştur (Rocker, 2006).

Algoritmik mimarinin temel yaklaşımı olan algoritmik düşünce, Xavier De Kestelier ve Brady Peters’a göre üretilen kodun sonuçlarını anlamak, yeni seçenekler keşfetmek için kodun nasıl değiştirilebileceğini bilmek, diğer tasarım potansiyelleri hakkında spekülasyon yapabilmek için yorumlayıcı bir rol üstlenmektir (Kestelier ve Peters, 2013). Mark Burry kodlama ve programlamanın iç içe bir konu olmasından ötürü, araçların kullanıcıları olan tasarımcıların, yeni araç yapımcıları, yazılım mühendisleri olduklarını düşünmektedir (Burry, 2011). Bu fikirle birlikte, Xavier De Kestelier ve Brady Peters da mimarların yazılım kullandıkları bir zamandan yazılım ürettikleri bir çağa doğru ilerlendiğini dile getirir (Kestelier ve Peters, 2013). Greg Lynn’e göre ise hala mimarın yazılım araçlarını nasıl geliştireceğini öğrenmesi, kod yazmayı öğrenmesi, tasarımı öğrenmesinden daha önemli değildir (Rocker, 2006). Aslen kodlamanın öğrenilmesini savunan isimlerin bu görüşe itiraz edeceklerini düşünemeyiz, çünkü işlemsel tasarım, araçları, kodlama ve programlamayı tasarımın işbirlikçisi olarak gören bir yaklaşımdır. Ancak bu görüşlere karşı fikirler de vardır. Mark Burry’in Scripting Culture kitabında tartışmaya açtığı ve önemli isimlerin görüşlerine yer verdiği konu ile ilgili John Frazer, kitabın basım tarihi 2011 yılı itibari

ile önümüzdeki 10 yıl içerisinde yaratıcı düşünce için betik yazmaya gerek duymadan yeni bir kavramsal çevreye sahip olacağımızı düşünmek istediğini ve betik ile elde edilen tek şeyin çizimi yazarak değiştirmek ise, hiçbir şeyin başarılmadığını dile getirmektedir (Burry, 2011). Tom Kvan da algoritmik anlayışın yaratıcılık üzerine etkisinin ne karanlıkta bir flaş, ne de cennetten bir nimet olmadığını, birçok alanda algoritmik olarak temsil edilen düşüncenin önemli ve kullanışlı olmasının yanında aslında münhasır olmadığını savunur (Burry, 2011). AKT Architects'ten Panagiotis Michalatos ve Sawako Kaijima betik bilgisinin yararlı ancak gerekli olmadığını, dijital medyaya ilgi ve yeteneği olmayan insanların farklı ve genelde ihtiyaç duyulan bir bakış açısı ve deneyimlerinin olabileceğini düşünmektedir (Burry, 2011). Aslen bu derece sert, karşı fikirlerin oluşmasındaki temel, kitabın yaratıcılık üzerine açılmış tartışma bölümünde bulunabileceği gibi, bugün algoritma tabanlı mimaride ortaya çıkarılan işlerin birbirlerinin tekrarı nitelikte, yanlızca kompleks formu hedefleyen yaklaşımda ve araçların yetkin bir şekilde kullanılamayarak başka kişilerin yazdığı program ve betiklere gebe işlerin görülmesinden kaynaklandığını düşünebiliriz. Robert Aish'in "..kontrolü ele almak isteyen bir tasarımcı da bir kod yazarı (scripter) olmalıdır (veya diğer insanların araçlarını kullanmanın görünmeyen etkisinin sonucundan muzdarip olmalıdır)" sözü tam da bu eleştirilerin sebebine ışık tutar niteliktedir (Burry, 2011, s.21'de atıfta bulunulduğu üzere). Tasarım ve programlama mesafesini Autodesk'in DesignScript dilini geliştirerek eriten Dr Robert Aish, tasarımcıların araçların kullanımındaki bu aksama ile alakalı olarak, dillerin daha kompakt, etkileyici ve anlaşılır hale geleceğini, kodlamaya odaklanmak için gerekli zihinsel çaba sebebi ile tasarım problemine odaklanılamayan bir dönemden sonra kodlamanın tasarımcılar için daha doğal, tasarım sürecine daha entegre bir hale gelip, artık dikkat dağıtıcılığının yok olacağını düşünmektedir (Burry, 2011).

Tüm karşı ve destekçi görüşlerin yanında, tasarımın kompleks süreci bir kenarda tutularak, betik ile formu çizme eylemine basit bir çerçeveden bakmak, aslen konunun özünde kalmayı sağlayacaktır. Paul Coates'in binada ilişkisel plan algoritması anlatımından yola çıkarak basit bir örnekleme yapılacak olursa; eğer bir tasarımcı tasarlayacağı masayı şöyle anlatırsa:

- bulunacağı oda metrekaresinin ... oranında
- odanın merkez noktasına ... pozisyonunda

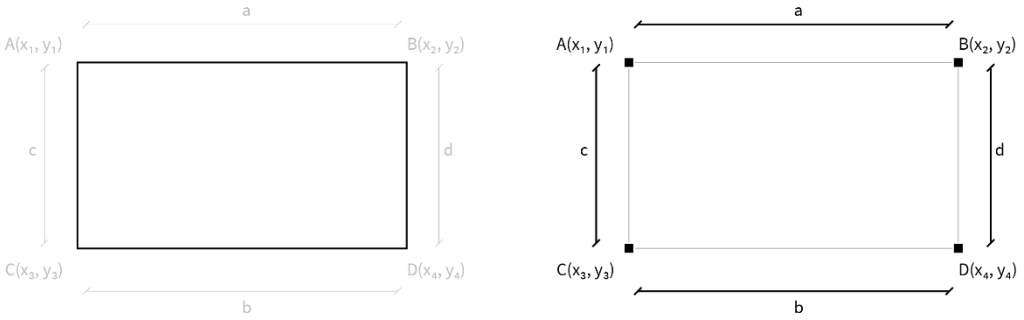
- bu pozisyondan ... yarıçapta

masa tasarlanmakla kalmayıp ilişkiler zinciri ile form oluşur, oda içi sirkülasyon çözümlenir, tasarım herhangi yeni bir odaya adapte olabilme özelliğindedir, çünkü form odadan gelen parametrelerle şekil alır. George Stiny ise değişkenlik üzerinden şu anlatımı yapar: "..hareketli A'lar ve X'ler, ancak çizdiğim çizgiler yok. Bu yeni bir çeşit paradoks mu? Şekiller statik olmalı ama benimkiler kararsız... Seçtiğim her ne ise görmekte özgürüm... Şekli nasıl çizdiğimin hiçbir önemi yok" (Stiny, 2006, s.4). Garry Stevens ise hipotenüs formülü ile algoritma mantığını açıklar: $c^2 = a^2 + b^2$ ise ve bu durum 3 – 4 – 5 de ve aslında daha birçok sayıda doğrulanabiliyorsa $c_x = a_x + b_x$ dir. Algoritmik mimarlıkta da bir kural seti oluşturulduktan sonra bu durumu sağlayan tüm varyasyonlara gitmek için açık uçlu algoritma ile formlar permütasyonuna gideriz (Stevens, 1990, s.21). Her iki anlatımda da karmaşık bir kural zinciri ve kompleks bir form önerisi yapılmamıştır, aksine basit bir dille ilişkiler ağı formüle edilmiştir. Aslen işlemsel tasarım, doku ve formlar deneyimleyerek anlamanın da ötesinde mantık ve prensiplerle anlaşılma noktasındadırlar (Tsigkari, ve diğ, 2011). Sean Ahlquist ve Achim Menges'in tanımı ile "statik biçimlenme"den "dinamik ilişkiler"e geçiş söz konusudur (Ahlquist ve Menges,2011).

Xavier De Kestelier ve Brady Peters'ın algoritmik mimarinin geleceği için görüşleri şu şekildedir; "Mimarların algoritmik kavramları yeterince anlamaları durumunda, dijitali farklı bir şey olarak tartışmaya artık ihtiyacımız olmadığında, hesaplama, mimarinin gerçek bir tasarım yöntemi haline gelebilecektir" (Kestelier ve Peters, 2013, s.15). Ancak bu süreçte önemli olan, Paul Coates'in Marvin Minsky'nin "Teorinizi sınamanıza ve sonuçlarını analiz etmenize yardımcı olan bir program yazmak ile sizin teoriniz olan bir program yazmayı birbirinden ayırmalısınız" sözleri üzerine değindiği; "Tasarımınızı sınamanıza ve sonuçlarını analiz etmenize yardımcı olan bir program yazmak ile sizin tasarımınız olan bir program yazmayı birbirinden ayırmalısınız" noktasıdır (Coates, 2010, s.26).

3.4.1.1 Kodlama ile form üretimi

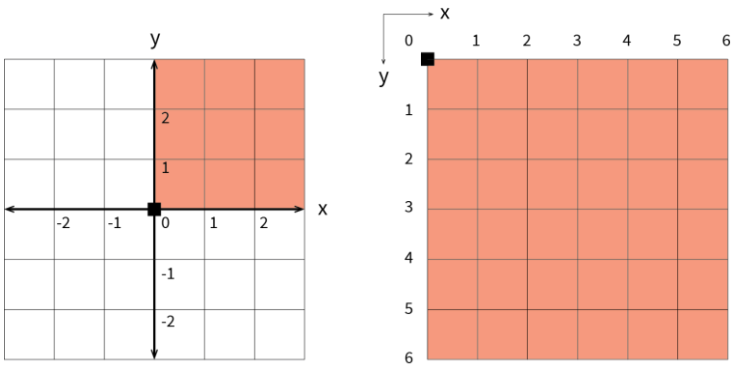
Bir formun kodlama ile ifadesinin niyeti, sayıların bir adım öne, çizgilerinse bir adım geriye gitmesidir. Başka bir ifade ile form, koordinatlarından uzunluklarına kadar tüm ilişkilerin sayılara ayrıştırılması ile oluşur (Şekil 3.16).



Şekil 3.16: Kodlamada form.

Programlama, Daniel Shiffman'ın tanımı ile bilgisayara takip etmesi için yönerge vermektir (Url-28). Bu açıdan ele alındığında kodlama için temel soru, nesnelere davranışlarını düzenleyen kuralları nasıl tanımlayacağımızdır (Shiffman, 2012). Her programlama dilinin kendi sözdizim ve alfabesine göre fonksiyonun yazılımı değişse de oluşma mantıkları aynıdır. Larry Cuba'ya göre bir programlama dili, bazı fikirleri ifade etme, diğerlerini ifade etme yeteneklerini sınırlandırma gücü verir (Reas ve Fry, 2007). Programlama, aslen bir yazma eylemidir ancak bir makale yazmakla arasında fark vardır (Reas ve Fry, 2007). İnsan dilinde, yazarın, kelimelerin belirsizliğini kullanması, dilde esneklik oluşturarak, tek bir metnin birden çok yorumuna el verirken, programlama dilinde yazmada, bir algoritma yazmanın birden çok yolu olması ve programlama dillerinin farklı gramerleri sebebi ile farklılık gösterse de kelimelerin belirsizliklerine yer yoktur (Reas ve Fry, 2007). Dolayısı ile insan dilinde, kurallara uygun bir cümle kurulmasa da, kelimeler eksik ve yanlış sıra ile kullanılsa da, kelimeler yanlış yazılsa da iletişim devam eder, ancak programlama dilinde, fonksiyonu anlatmak için bir yazım yanlışı yapılır, bir *","* veya bir *")"* veya bir *";"* sembolü unutulursa dahi, bilgisayar ile iletişim kurmaya devam etmenin bir yolu yoktur.

Casey Reas ve Ben Fry'ın 2001 yılında programlama ve görselleme dilini bir araya getirdikleri, programlama sözdizimine hemen hemen herkesin kolay bir şekilde adapte olabileceği Processing dilinde, genel olarak formun betik dilinde temel oluşumunu incelemek mümkündür. Öncelikli olarak yazının yazılacağı kağıda, bir kanvasa ihtiyaç vardır ve bu kanvas koordinat sistemi ile kartezyen düzlemi tanımlaması ile yapılır (Şekil 3.17). Bilgisayar ekranında oluşturulan bu kanvasın kartezyen düzleminden tek farkı (0,0) noktasının düzlemin merkezinde değil sol üst köşesinde oluşmasıdır. Bu durumda x ve y doğrultuları için negatif değerler yoktur (Url-29).

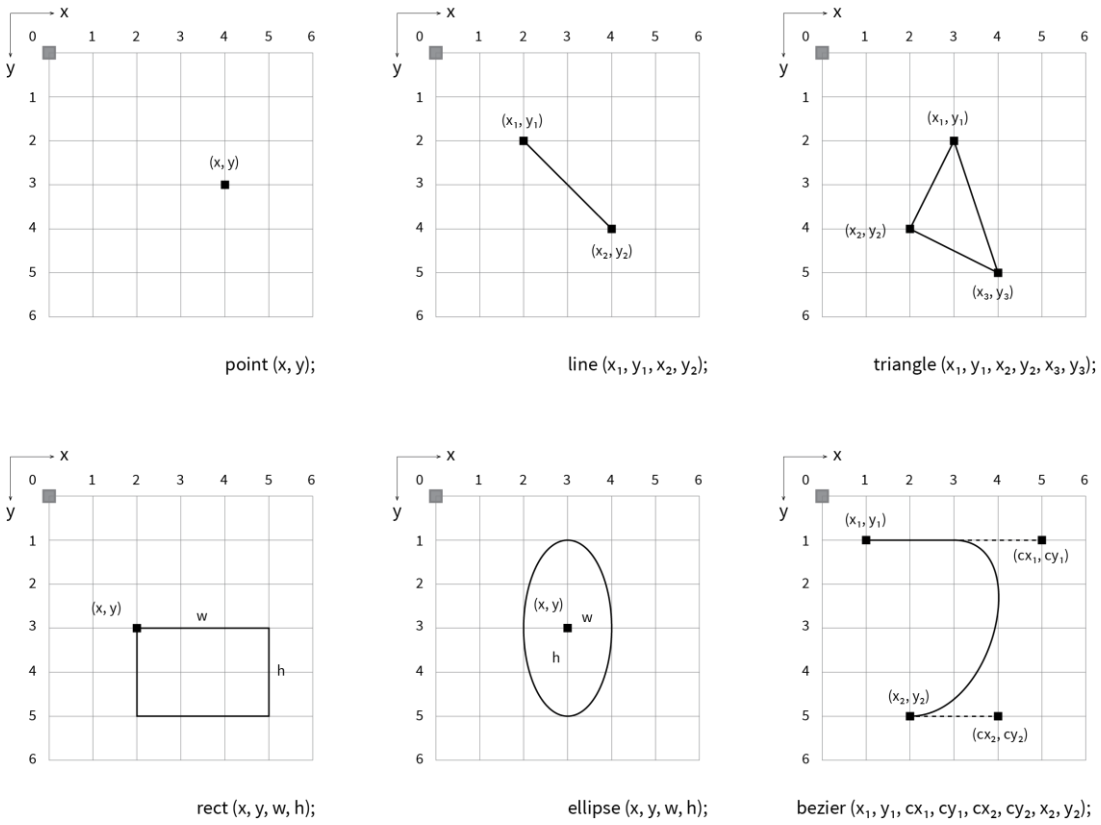


Şekil 3.17: Daniel Shiffman kartezyen düzlemi-kanvas anlatımı (Url-34).

Kanvas (x,y) limitleri belirlenerek oluşturulur ardından çizim bu kanvas içerisinde yapılır. Çizilecek eleman kod içerisinde fonksiyon olarak adlandırılır(Şekil 3.18). Aslen fonksiyonlar, şekiller çizmeye, renk ayarları yapmaya, sayıları hesaplamaya ve diğer pek çok eylemi gerçekleştirmeye olanak tanır (Reas&Fry, 2007). Bir işlev genelde bir kelimeden oluşup ardından parantez gelir ve parantez içerisinde de virgül ile ayrılmış öğelere parametre denir, parametrelerin sıralanışı ve formülü ise dilin sözdizimine göre değişkenlik gösterir (Reas ve Fry, 2007).

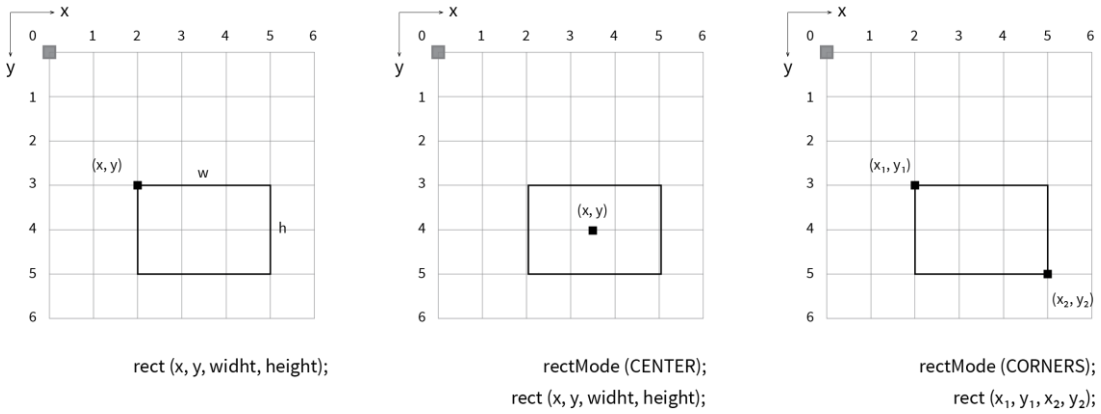
Şekil 3.18: Processing sözdizimi.

Her çizimde, şeklin yer, boyut, renk gibi tanımlanmak istenen değerlerini oluşturmak için gerekli bilgileri ve dilin bilgiyi nasıl alacağını bilmek gerekir (Url-29). Buna göre temel çizimler şu şekilde oluşturulur (Şekil 3.19).



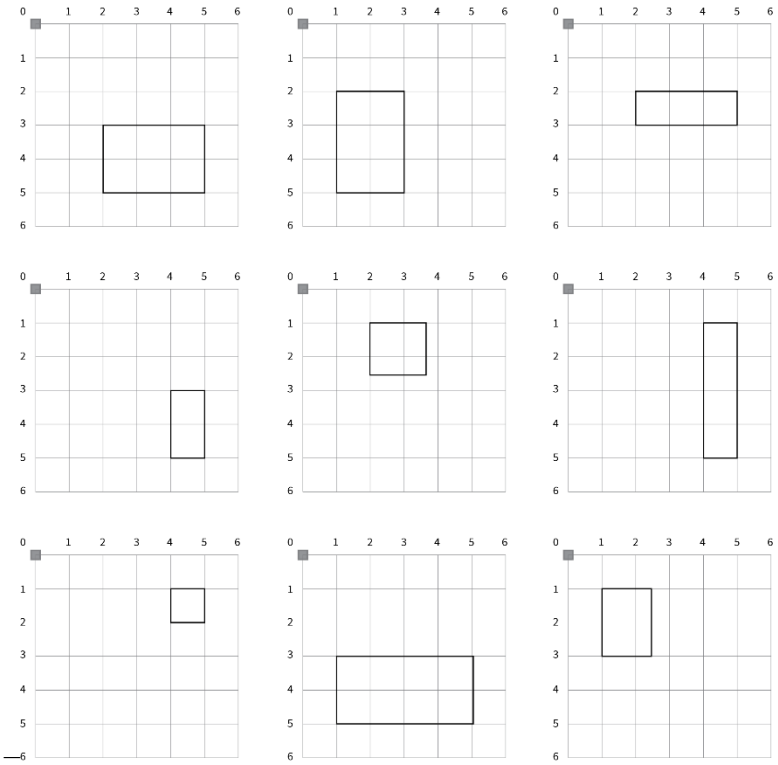
Şekil 3.19: Processing temel çizimler.

Bir şeklin aynı zamanda birden çok yazımı mevcuttur (Şekil 3.20).



Şekil 3.20: Processing dikdörtgen fonksiyonları.

Nihai fonksiyonlar ile, oluşturulan kanvas limitlerinde, elde edilebilecek pek çok sayıda olasılık tanımlanmış olur (Şekil 3.21).



Şekil 3.21: Processing dikdörtgenler olasılıkları.

(6,6) pixel kanvas üzerinde oluşturulabilecek dikdörtgen sayısı dahi yüzlercedir. Bu sayı yanlızca dikdörtgenin kenar uzunlukları ve düzlemdeki pozisyon değişkeni ile bile oldukça fazla iken formun oluşmasına etki eden diğer ilişkiler birer değişken olarak tanımlandıktan sonra, keşfedilmeyi bekleyen sonsuz elemanlı olasılıklar kümesi oluşur. Richard Dawking'in sözleri formun betik ile sonsuzlaşmasını şu şekilde anlatır; "Ne bir biyolog olarak önsezilerim, ne programlamadaki 20 yıllık deneyimlerim, ne de çılgın hayallerim, beni ekranda belirenlere hazırlamıyor" (Reas ve McWiiliams, 2010, s.161).

3.4.1.2 Formun matematiksel tanımı

"Mathematik ve mimari arasındaki ilişki çok ve çok önemli, eğer matematik icad edilmemiş olsaydı, mimarlar kendileri icad etmek zorunda kalacaklardı" (Salvadori, 2015, s.25).

Formun betik dilinde tanımlanması ile geometrik ve matematiksel bir anlatım, sürecin olağanlaştırdığı bir gereklilik olur. Form bu sayede "..şekillenenden ziyade hesaplanan, çizilenden ziyade yazılan" olmaktadır (Legendre, 2011,, s.11). "hesap" ve "yazı" "algoritma" ve "kurallı dil" kavramlarına karşılık gelmektedir. Aslen bu iki kavram ile oluşturulan formun matematiksel tanımı yapılmaktadır. Bir diğer açıdan

bakılacak olursa formun matematiksel tanımı, formun hesaplanabilir ve yazılabilir olmasını sağlamıştır. Başka bir ifade ile matematiğin ve betik dilinin doğası içerisinde form da kurallı bir doğaya bürünmektedir. Mario Salvadori Mathematics in Architecture kitabında matematik ve mimari arasındaki ilişki için şu örnekleri verir; bir kenarı 1, diğer kenarı x olan bir dikdörtgenin alanının x değerlerine göre değişimini bulalım. Bu integraldir. Integral 0 ile x arasında $dx= 1*x=x$ olmak üzere $f(x)=x$ dir. Bu tanımlamayla matematik, ötede değil, problemi çözümlemek için pratik bir değerdedir (Salvadori, 1968). Bir başka örnekde bir dikdörtgenin alanını hesaplamak istersek, bir kenarı x diğer kenarı y olan dikdörtgenin alanı $x*y$ olarak ifade edilir. $y= 50-x$ ise $alan= x*y = x*(50-x) = 50x-x^2$ olup x değerleri değiştikçe alandaki değişim önce artan sonra azalan bir eğridir. Bu dikdörtgenin x ve y si arasındaki ilişkisel anlatım, sadece belirlenen x değerlerine, x'in değerlerinin belirleneceği aralığa göre sonuç üretmeyi sağlamamış, x ve y'nin ilişkisindeki değişime göre de sonuçlar üretilebilir olmuştur. x'in hangi değerinde dikdötgen alanı maksimum, hangi değerinde minimum olur sorusunun cevabı, x ve y nin ilişkisinin sonuç değerine doğrudan etkisini gösterir. Birden çok dikdörtgenin yanyana kurulmuş ilişkisel anlatımı ise başka ilişkisel sonuçlar ortaya çıkarır (Salvadori, 1968). Salvadori'nin örneklerinde görüldüğü gibi bir kenarı 3 bir kenarı 5 olan dikdörtgen tanımının ötesinde, bir kenarı x diğer kenarı x+2 olan dikdörtgen tanımlamasına geçilip, farazi sayılarla değil, kurala tabi bir dikdörtgen tanımı, aslen dikdörtgenin tanımını kuvvetlendirmekle birlikte tanımın aralığını genişletmektedir. D'Arcy Wentworth Thompson'a göre de formun matematiksel tanımı, yanlızca tanımlamamızın daha önceki aşamasında eksik olan bir hassasiyet kalitesine sahiptir ve bu sayede, daha önce açıkça görülen homoloji veya kimlikler keşfedilir; açıklamalardan ziyade açıklığa kavuşmuş açıklamalar keşfedilir (Reas ve McWilliams, 2010, s.75). Thompson'ın altını çizdiği nokta, matematiksel tanımlama ile birlikte, formun tanımı yeni bir keşif değil, mevcudu geliştirmek ve genişletmektir.

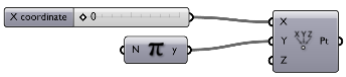
Bu yaklaşıma karşı görüşler mevcuttur. Cecil Balmond bir manifesto için bir temelin olmadığı zamanlarda olduğumuzu, geometrinin, stilin, dilin başvurulmadığı, aranmadığı, unutulduğu bir dönemde olduğumuzu savunurken, tasarımda matemetiğin rolü için Paul Coates Victoria döneminden kalma bir kalıntı olduğunu düşünür (Burry, 2010, s.62'de atıfta bulunulduğu gibi). Ancak bugün oldukça paradoksal bir durum söz konusudur. Antoine Picon'un desteklediği üzere dijital araçların etkisi altındaki

mimari, "..bezier eğrilerinden algoritmalara kadar, matematik ile ilişkisi olduğunu hiç düşünmeden pek çok matematiksel obje kullanmaktadır" (Picon, 2011, s.30). Aslen işlemsel tasarım da matematiğe yeni veya keşfedilmemiş olarak bakmamaktadır. Matematik ile, işlemsel tasarımda formun keşfi değil yeniden keşfi, yeni tanımı değil yeniden tanımı yapılmaktadır. Mark Burry, Scripting Culture kitabında Flann O'Brien'nin kültleşmiş kitabı Üçüncü Polis'den bir alıntı yapar; "Joe bu esnada birşeyler açıklıyordu. Yine, tanıdıklarının yeniden keşfedilmesinin, bitmemiş olanın başlangıcı olduğunu söyledi. Zaten acı çekenlerin yeniden deneyimi, unutulmayan hazinenin unutulması gibidir.." (Burry, 2010, s.222). Matemetik ve formun ilişkisi Joe'nun anlattığı zıtlıkları barındıran ilişkiler gibidir. Matematik, bitmemişin başlangıcı, tanınan ile yeniden tanışmak, çoktan yaşananın yeniden deneyimlenmesi, unutulmayanın unutulması gibidir. Form matematikten ayrı dahi düşünülemez ve aslen hiçbir zaman ayrılmamış ilişkilerine bugün yeniden ve başka bir gözle bakmaya, yeniden deneyimlenmeye çalışılmaktadır. Kurallı bir dilin çerçevesinde yeniden tanımlamak ve formu yeniden keşfetmek, formun daha önce yüz üstünde olup bugünkü tanımlaması ile daha da netleşen potansiyelleri ile yeniden tanışmaktayız.

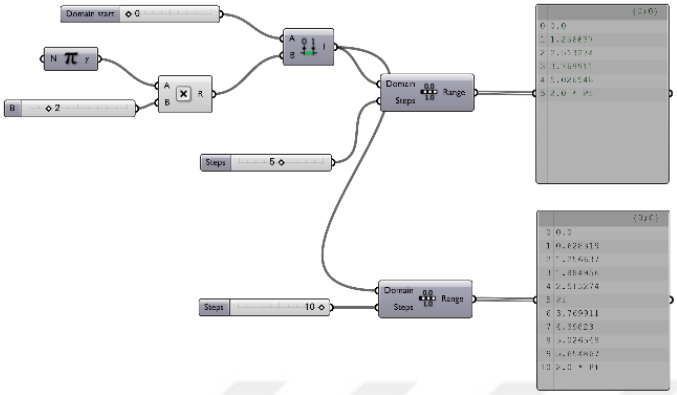
Kodlama ile geometri oluşumu için Joseph Choma'nın Morphing kitabında temel geometrilerin matematiksel formül karşılıklarından, kompleks geometrilerin formüllerine adım adım inceleme yapılmıştır. Bu formüllerden bazıları kullanılarak, David Rutten tarafından geliştirilmiş Rhino'nun Grasshopper eklentisinde geometri çizimleri tarafımca bu teze eklenmiştir.

Grasshopper kanvasında Processing dilinde olduğu gibi hayali bir düzlem vardır. Ancak Grasshopper Rhino çizim programı ile senkronize olarak çalışıp, oluşturulan betiğin görsel karşılığı Rhino düzleminde oluşmaktadır. Dolayısı ile Rhino'nun sonsuz uzayı Grasshopper'ın da kanvasıdır ve Processing'de olduğu gibi kanvas tanımı yapılmaz. Fakat Grasshopper dilinde değişkenlerin aralıkları diğer kodlama dillerinde olduğu gibi önem taşımaktadır.

Buna göre geometri değişkenlerine sabit sayılar tanımlanabilmekle birlikte (Şekil 3.22), değişken değerleri için sayı değerleri aralığı tanımlamak, forma etki edecek bu aralık içerisindeki tüm değerlerin önceden sınırlarını oluşturmaktır. 0 ile 2π arasında irrasyonel, rasyonel, tüm sayıları düşünecek olursak, pek çok sayıda değer mevcuttur. Ancak bu aralıktaki değerlerin aralığını belirlemek de mümkündür ve bu da değişken değer aralığı içerisinde ikincil bir filtreleme sağlar (Şekil 3.23).

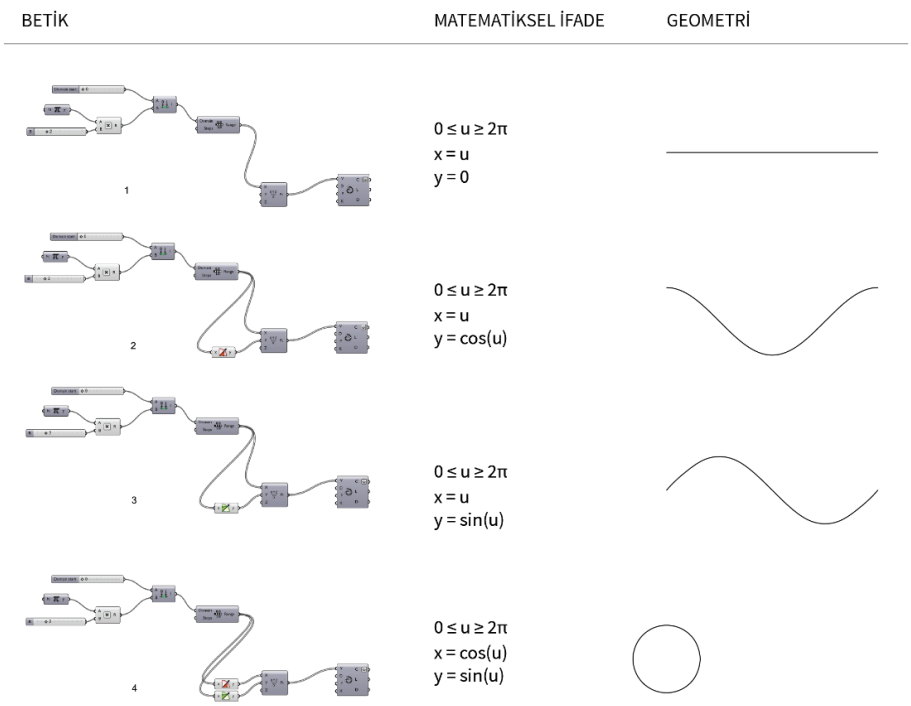


Şekil 3.22: Sabit değerde değişken tanımı.



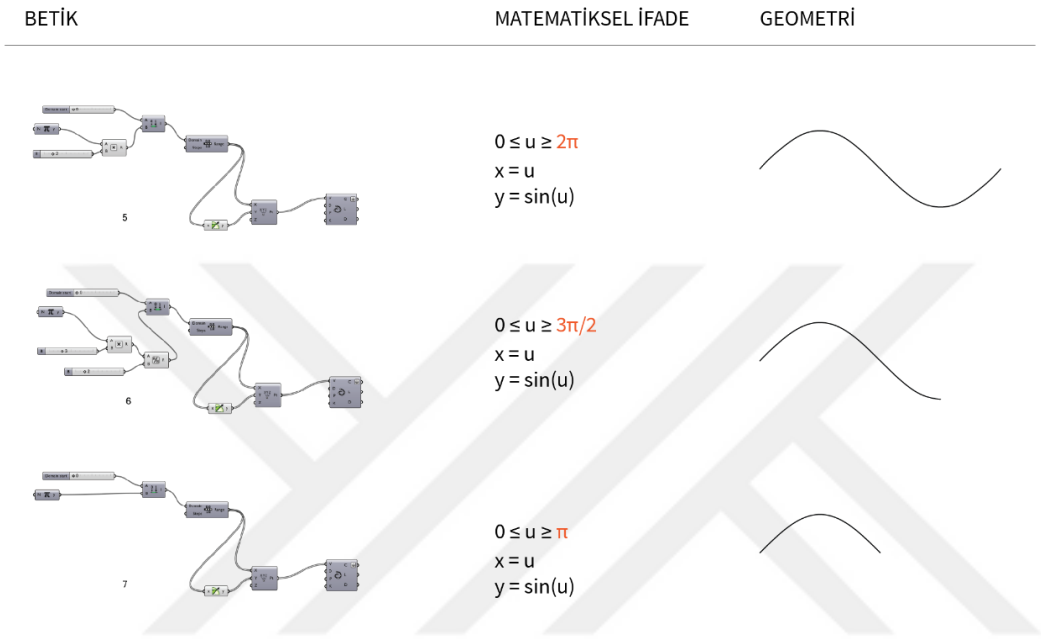
Şekil 3.23: Değişken değerler için değer aralığı tanımlama.

Buna göre bir çizgi, x, y, z değişkenlerinden ikisi sabit tutulmak üzere belirlenmiş bir değer aralığının tek bir değişkene uygulanması ile elde edilir. Eğri çizimi için sinüs ve cosinüs, istenilen yönde eğri oluşacak şekilde x, y, z değişkenlerine uygulanır. Bir daire çizimi ise zıt yönde eğriler oluşturan sinüs ve cosinüsün birlikte kulanılması ile elde edilmektedir (Şekil 3.24).



Şekil 3.24: Grasshopper temel geometri çizimleri.

Temel geometrilerin oluşturulmasının haricinde dönüştürme (transform), işlemleri yine matematiksel formüller ile çözümlenebilmektedir. Bazı dönüşüm işlemleri içinden örnek vermek gerekirse, kesme işlemi değişken değer aralığının değiştirilmesi ile gerçekleşir (Şekil 3.25). Bu sayede şekilde görülebileceği gibi 0 ile 2π arasındaki değerlerin eğrinin tamamlanmış halini oluşturduğu kabul edilirse 0 ile π arası, eğrinin yarısını oluşturacaktır.



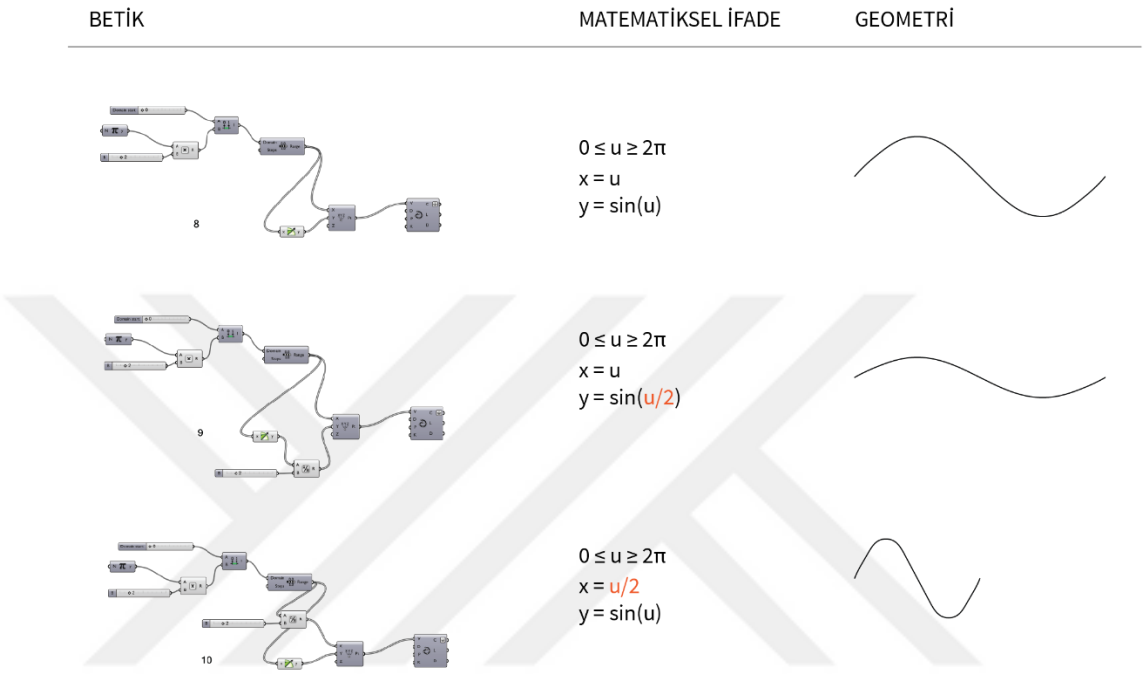
Şekil 3.25: Kesme.

Ölçek (scale) işlemi, değer aralığı aynı kalırken aralık içindeki değerlerin katsayıları değiştirilerek yapılır. Buna göre 0 ile 2π aralığındaki değer u iken x = u ve x = 2u, x'in değerlerini 2 katına çıkarır ve bu ölçek olarak formda etkisini gösterir (Şekil 3.26).

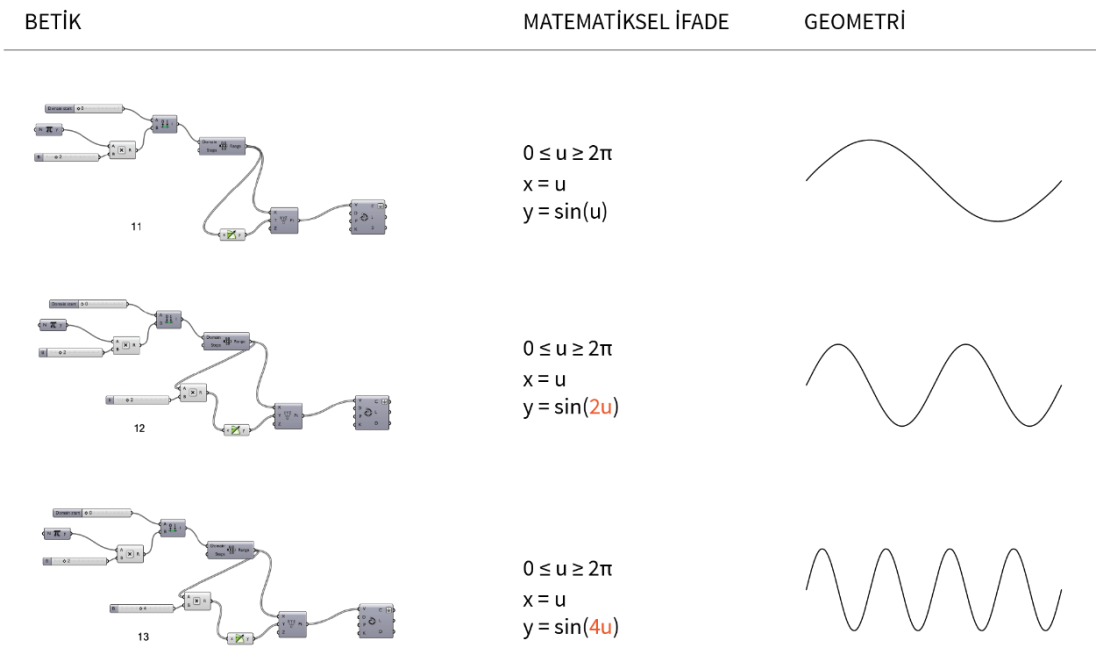
Modül oluşumu aslen tekrarlama (repetation) olarak düşünülebilir. Bunda sinüs veya cosinüs değerlerinin aynı değer aralığından alınacak farklı katsayıdaki değerleri ile, her zaman 0 ve 1 aralığında sonuç veren sinüs ve cosinüsün, katsayı etkisi ile tekrar oluşturmasına sebep olmaktadır. Bu durum katsayı artışı ile 0 ve 1 arasında daha çok aynı sonuç elde edilip tekrar eden modüller oluşur (Şekil 3.27).

Sıkıştırma (pinching) işlemi ise yine modülleme mantığı ile, sinüs ve cosinüs değerlerinin farklı katsayıları ile oluşturulur, ancak bu işlemdeki etki modüllemeden farklı olarak, gittikçe artan veya azalan farklılaşma olması için daha büyük katsayı farkları oluşturmak adına değerlerin karesi, küpü gibi üslü değerleri alınmıştır (Şekil 3.28).

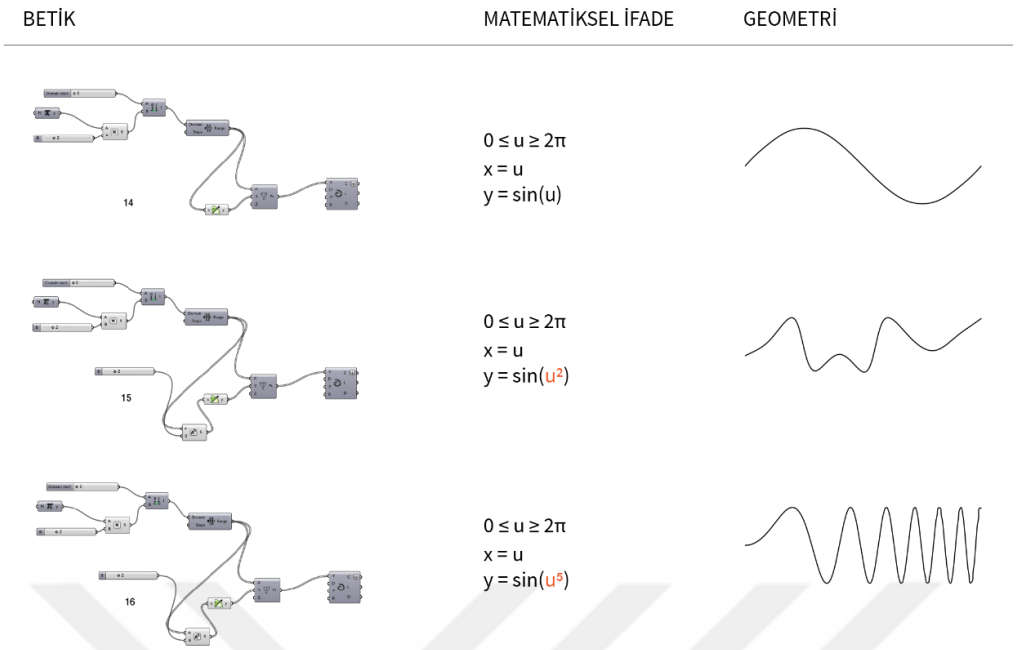
Aslen tüm bu dönüştürme işlemleri çalışmalarından sonra matematiğin iki seviyesinden söz edebiliriz. Birincisi formu meydana getiren formül seviyesi, ikincisi ise dönüştürme etkisi oluşturan katsayı değişimi seviyesidir. Birinci seviyede formu oluşturan noktalar arası ilişkiler tanımlanırken ikinci seviyede değişkenler, değerler arası matematiksel ilişki tanımı yapılır.



Şekil 3.26: Ölçekleme.



Şekil 3.27: Modülasyon.



Şekil 3.28: Sıkıştırma.

Örneklerde görüldüğü üzere, aslen betik dili içerisinde form, matematiksel bir yapı içerisinde oluşmaktadır. Gehry Technologies'den Nick Pisca betik ve betik dili ile tanımlanan form için matematiğin temel olduğunu şu sözleri ile aktarır; "Eğer mantığı matematiğin bir alt kümesi olarak görürseniz, o zaman matematik esastır. Ortalama bir kod çözücünün tek bilmesi gereken, basit bir betik oluşturmak için cebir, 3 boyut için ise tirgonometridir" (Burry, 2010, s.61'de atıfta bulunulduğu gibi). Yalnızca matematiksel fomüller ele alındığında oluşabilecek geometrik formları tahmin etmek zor iken, yanlızca geometriyi ele almak da formun nasıl oluşturulabileceği hakkında ipucu vermemektedir. Yanlızca tek bir katsayı değişiminin iki form arasındaki fark olduğunu bilmek oldukça güçtür. Formun matematik ile açıklanması geometridir. Geometri, problemlerin görselleştirilmesine olanak tanırken, cebir, cevapların niceliksel değerlendirmesini basitleştirir (Salvadori, 1968). Mario Salvadori geometri ve cebir ilişkisini Descartes ile anlatır. Geometri ve cebiri bir araya getirerek analitik geometriyi icad eden Descartes'in, geometri ve cebirin aynı soruna bakmanın iki yolu olduğunu gösterdiğini dile getirir (Salvadori, 1968).

Sonuç olarak formu kurallı hale getirmek, tanımının matematiksel bir dile dönüşmesi ile mümkün olmaktadır. Bu da nihai bir form değil, kuralları sağlayan tüm değerleri kapsayan bir tanımlama yapmaktır ve aslen sonsuz sayıda olasılık aralığı oluşur. Bu bağlamda sonsuz olasılık aralığı anlatımını açmak için yine bir uygulama üzerinden

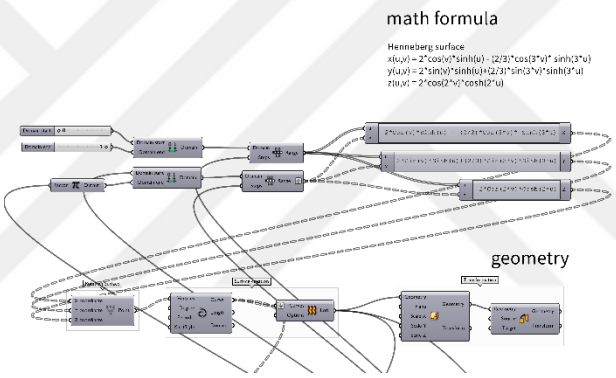
ilerlenebilir. Uygulamada matematik formülü olarak minimal yüzeylerden Henneberg yüzeyini tarifleyen matematiksel formül kullanılacaktır. Minimal yüzeyler çalışması, Plateaus'un "belirli bir eğriyi kapsayan en küçük alanı içeren yüzeyi bulma" problemi ile başlamıştır (Url-30). İlk olarak Meusnier'in katenoid ve helikoidi 1776 yılında bulması üzerine çalışmalar gelişmiştir (Url-31). Hennenberg yüzey fomülü şu şekildedir:

$$x(u,v) = 2*\cos(v)*\sinh(u) - (2/3)*\cos(3*v)* \sinh(3*u)$$

$$y(u,v) = 2*\sin(v)*\sinh(u)+(2/3)*\sin(3*v)*\sinh(3*u)$$

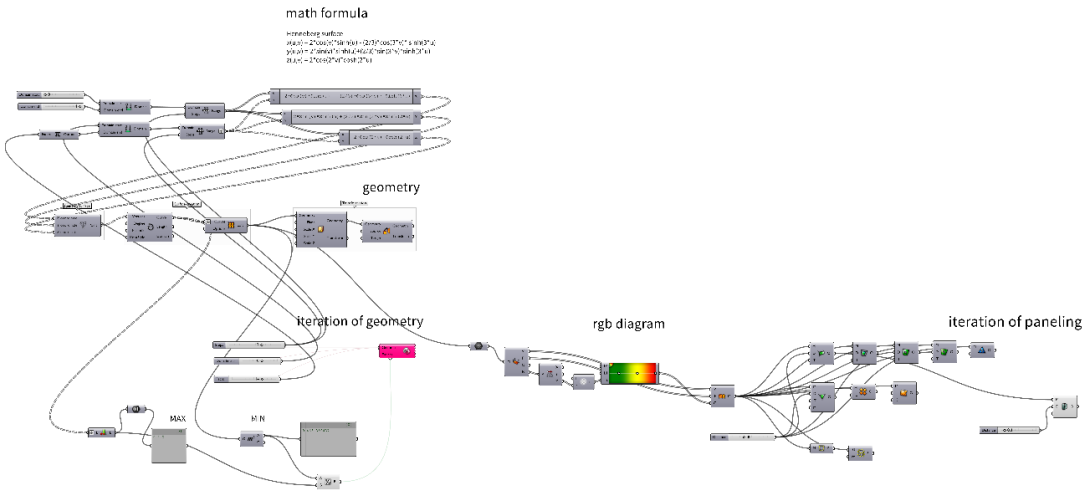
$$z(u,v) = 2*\cos(2*v)*\cosh(2*u)$$

Öncelikli olarak bu formül üzerinden Grasshopper kodlaması ile yüzey oluşturulmuştur (Url-32). Ardından varyasyonlar, üretilmiştir (Şekil 3.29).

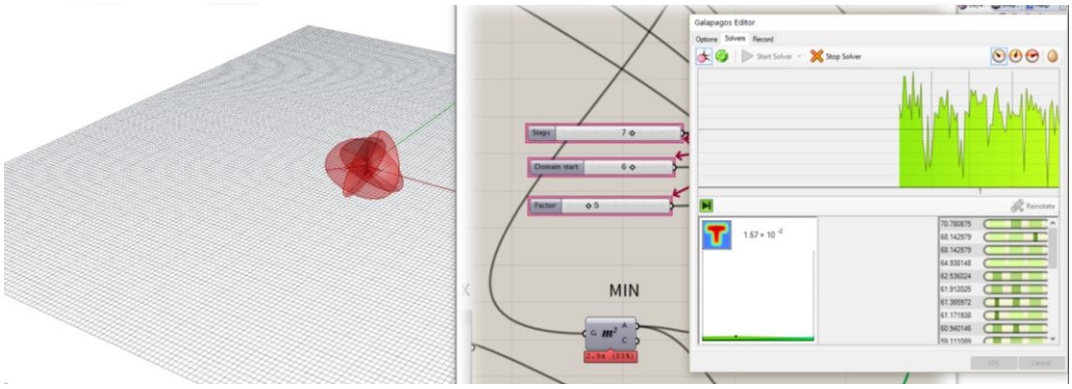


Şekil 3.29: Hennenberg yüzey kodlaması

İlk olasılık aralığı geometrinin kendi değişkenleri üzerinden belirlenen kurala göre üretilmeye başlanmıştır (Şekil 3.30). Kural, maksimum sayıda yüzeyi oluşturan eğri ile minumum yüzey alanı sağlanması olarak belirlendikten sonra optimum sonuç elde edilene kadar adım adım form olasılıkları oluşturulmuştur (Şekil 3.31-3.32).



Şekil 3.30: Grasshopper kodu.

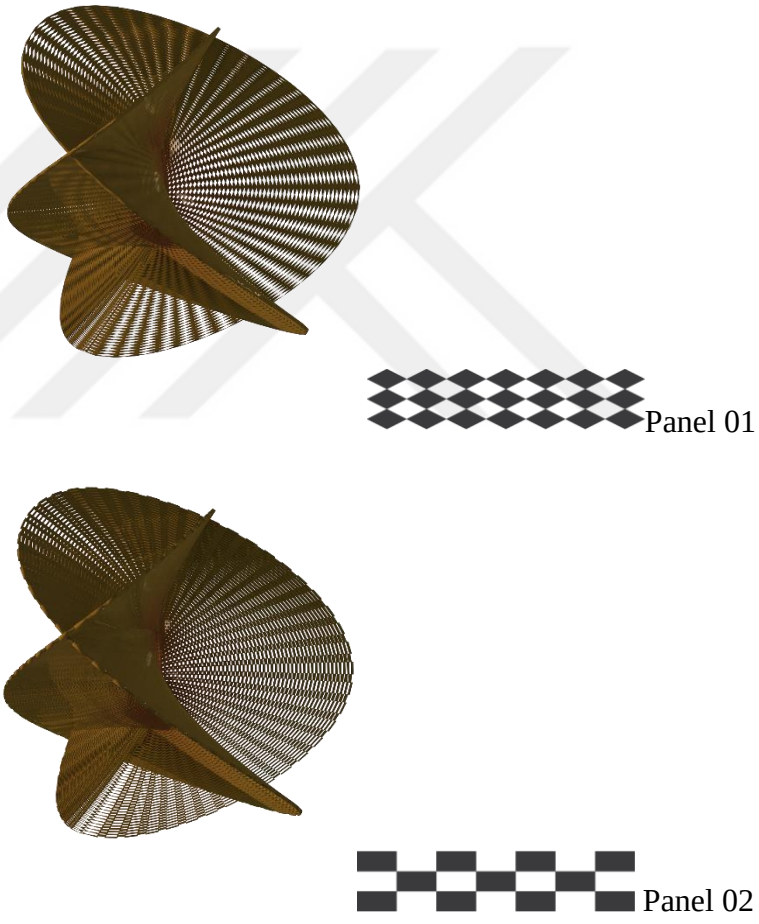


Şekil 3.31: Genom oluşumu.

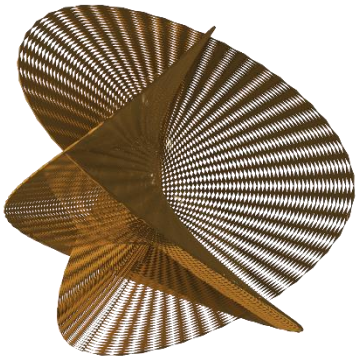
Şekil 3.32: Varyasyon üretimi.

Maksimum sayıda eğri ile minumum yüzey alanı sağlayan form üretilen varyasyon sınırı içerisinde belirlenip, seçilen bir form üzerinden panelleme çalışması ile ikincil varyasyon kümesi oluşturulmuştur (Şekil 3.33-3.34).

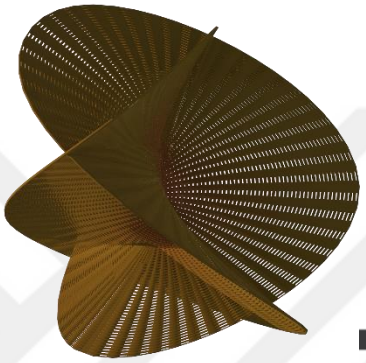
Şekil 3.33: Maksimum sayıda eğri ile minumum yüzey alanı sağlayan form.



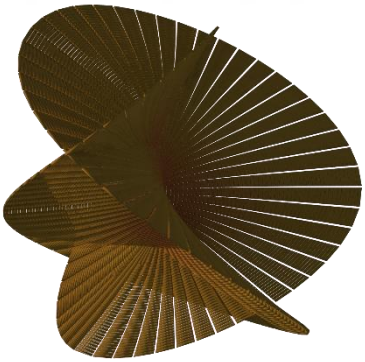
Şekil 3.34: Panel tipleri.



Panel 03



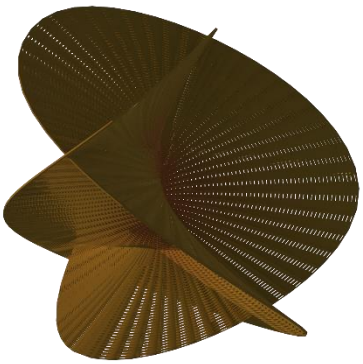
Panel 04



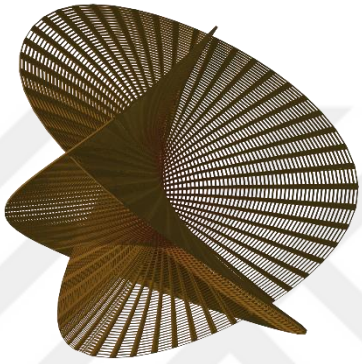
Panel 05

Şekil 3.34 (devam): Panel tipleri.

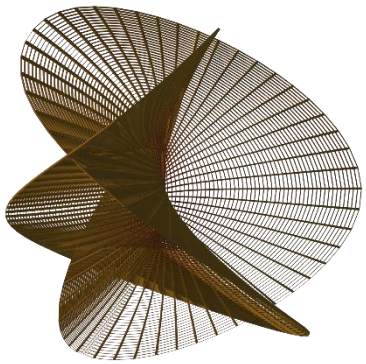
Son adımda panel 04 kullanılarak, farklı panel boyutları denemesi yapılıp bir kez daha varyasyonlar çıkarılmıştır (Şekil 3.35).



Ölçülendirme01



Ölçülendirme02



Ölçülendirme03

Şekil 3.35: Panel tipi 04 farklı panel ölçüleri ile varyasyon oluşumu.

Uygulamada öncelikli olarak 21 farklı form, daha sonra belirlenen 1 form üzerinden 5 farklı panel tipi ve son olarak 3 farklı panel boyutlaması ile 3 adımda varyasyonlar oluşturulmuş toplamda 29 tip form üretilmiştir. Üretilmiş 21 formun her biri için 5 farklı panel tipi uygulaması yapılsa idi elde edilecek varyasyon sayısı 105 olacaktı. 105 varyasyona 3 çeşit panel boyutlaması yapılması durumunda bu sayı 315'e ulaşacaktı. Bu sayının yanlızca 5 farklı panel tipi ve 3 farklı boyutlandırma ile oluştuğu

düşünülürse, yalnızca 10 tip panelleme ve boyutlama ile varyasyon sayısının 2100'e ulaşacağı, ilk varyasyon oluşturma adımında form sayısının 21'den 100' çıkarılması durumunda da sayının 10000'e ulaşabileceği hesap edilebilir. Aslen yalnızca sınırlı sayılarda parametrelerle dahi oluşacak varyasyon sayılarının oldukça fazla sayıda olduğu hesaplandığında, sonsuz olasılık aralığı ifadesinin kullanım sebebi netleşmiştir.

4. TASARI

“Doğa bize kendini bir kargaşa olarak gösterse de, doğayı dirilten düzendir” Le Corbusier (Schumacher, 2017, s.18’de atıfta bulunulduğu gibi).

Form bir süreç ve bir sistemin ürünüdür. DeLanda nesnenin var oluş nedenini, süreçlerin var olması ile açıklar (DeLanda, 2009). Form üretiminde süreç, formu oluşturacak sistemin kurgulandığı lineer olmayan bir zamanı tarifler. Aslen bu sürecin lineer olmaması, işlemsel tasarım sisteminin gerekliliğini, yanlızca kompleks formların geleneksel yöntemler ile yapılamamasından dolayı değil, Robert Aish’in savunduğu üzere, sürecin kontrolünün büsbütün sağlanma uğraşısından dolayı olduğunu gösterir. Süreçte form oluşumunu ise D’arcy Thompson, bazı matematik kuralları takip ederek örüntüler şekillendiren, iç ve dış kuvvetler altında kendini yeniden şartlara göre organize eden bir sistem olarak yorumlar. Esasen bu sürecin kontrol ve sistemleştirilmesi, işlemsel tasarım düşüncesinde cevabını bulmaya başlamıştır, limitleri zorlamanın ötesinde bir gereklilik halini almıştır (Khabazi, 2010). Form bu süreçte yerini, dondurulmuş bir an olarak değil, bir zaman olarak göstermekte, deneyimlenmekte ve keşfedilmektedir. Tasarımda hesaplama artık yanlızca bir araç olmanın da ötesinde, tasarımcının düşünce yapısının değişmesinin kaçınılmaz olduğu bir ortam olarak hızla olgunlaşmıştır.

Tartışılmış kavram ve fikirlerin, dijital ortamda somut modellere dönüşmesi değerli görülmektedir. Aslen bu bölümde bir form oluşum sürecinin yansıması yapılmak istenmiş ve bu doğrultuda ele alınmış düşünce yapısı ile süreç deşifrelenmeye çalışılmıştır. Tasarım süreci tartışmalı bir konu olmakla birlikte farklı yaklaşım ve fikirleri bünyesinde barındıran bir süreçtir. Ancak çalışmanın form üretimi olmasından çok form arama denemeleri olarak okunması ve tez kapsamında ele alınmış işlemsel tasarımın matematiksel düşünüş yapısı üzerinden değerlendirilmesi doğru olacaktır. Tasarım girdilerinden yanlızca matematik aracılığı ile kurallı geometrinin oluşturulması konu edilmiştir. Dolayısı ile tanımlaması yapılacak form oluşum sürecinde tez strüktürünü oluşturan akıl ve dil yapısına odaklanılmalıdır. İşlemsel tasarım düşüncesi ile form oluşturma, form için sonsuz olasılık aralığı oluşturmuştur.

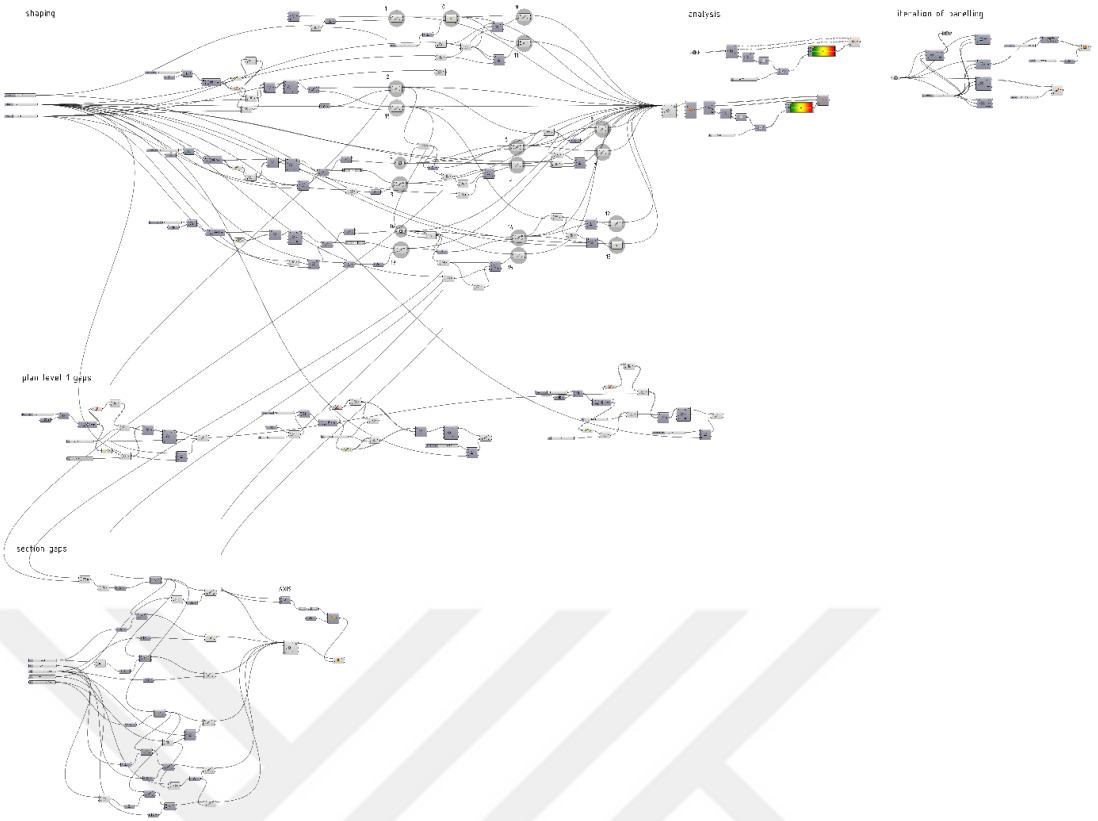
Sabit değer ve ilişkilerin tanımlanmadığı sistem sayesinde tasarımcının zihninde tasarladığı formu dijital araçlarla geometrik olarak ifade etmesinden öte, geri bildirim aldığı, süreç içerisinde karar verdiği ve sürekli denemeler yaptığı bir süreç oluşmuştur. Yanlızca değişken değerler ile ilişkilerin tanımlandığı geometriler düşünüldüğünde bile, değer ve ilişkilerin yeniden tanımlanması ile nihai olacak form sürekli denenen, keşfedilen bir zaman aralığı içerisinde dondurulmuş bir an olacaktır.

Aslen programların tasarımcıların programlama, tasarımda algoritma geliştirme konularına yaklaşmasında ciddi bir önemi oluşmaktadır. Tasarımcıların birincil olarak kodlamanın yoğunluğunda kaybolmadığı, görsel ilişki kurabildiği ve basit bir arayüzde keşfedebildiği bir program, kullanımı arttırmaktadır. Görsel programlama dilleri bu sebeple tasarımcıların daha fazla tercih ettiği programlar olmuştur. Dijital ortam olarak bu çalışmada Rhino Grasshopper eklentisi kullanılmıştır. Tasarımcılar için Rhino arayüzünde eş zamanlı olarak algoritma ile oluşan geometrinin karşılığını görmek ve Grashopper'ın kullanıcılar için basit arayüzü programın tercihinin temel sebepleri arasında gösterilebilir.

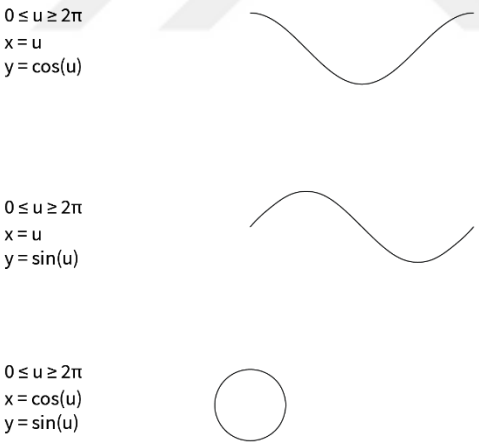
4.1 Matematiksel Dil İle Geometri Oluşturma

Çalışmada kabuk tasarımı, zemin ile ilişki, cephelenme, yüzey çalışmaları, strüktür oluşturma gibi pek çok konuyu içerisinde barındırması sebebi ile form çalışma alanı olarak belirlenmiştir. Süreç adımlara bölünmüş, ilk olarak form arayış süreci algoritmalar ile 2 boyutlu düzlemde kapalı bir eğri oluşturma ile başlamış daha sonra tekrar geliştirilen eğri analizi algoritmalarından geri bildirim alınarak 3. boyuta geçiş yapılmıştır. Ardından strüktür oluşturulmuş. Daha sonra yüzey panellemeleri için panel sayılarını optimize edebilen bir algoritma geliştirilmiştir. Tüm adımlar da kendi içlerinde varyasyonlar barındırıp nihai forma, algoritmalar ile adımlar serisi sonunda ulaşılmıştır (Şekil 4.1).

Öncelikli olarak bir önceki bölümde matematik formülleri ile yapılmış çalışmalar neticesinde öğrenilmiş geometri algoritmalarının kullanımı ile 2 boyutlu kapalı bir eğri oluşturulmuştur (Şekil 4.2).



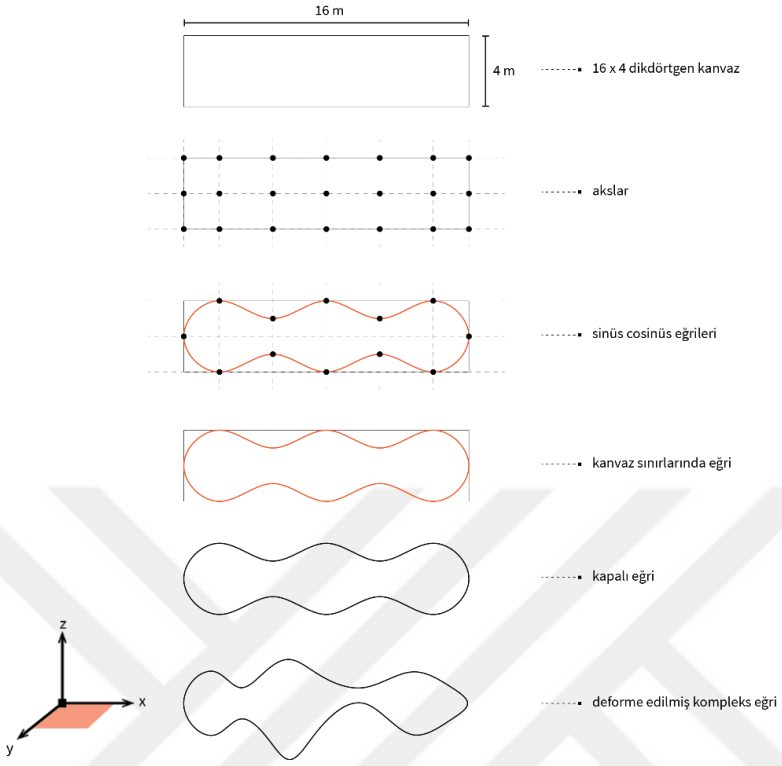
Şekil 4. 1: Geometri algoritması.



Şekil 4. 2: Geometri oluşumunda kullanılmış temel formüller.

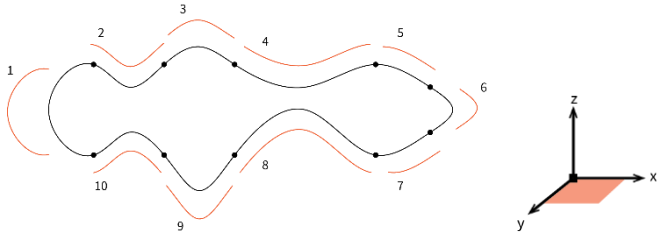
2 boyutlu eğrinin oluşumunda aslen bir dikdörtgenin kenarlarının deformasyonu fikri ile 16nx4n metre ölçülerinde, bir dikdörtgenin sınırlarına tabi olarak tasarıma başlanmıştır. Matematik formülleri kullanılarak kapalı geometri oluşturulurken, sinüs ve cosinüsün zıt yönde eğriler oluşturması doğası gereği, formüller ile yeniden tanımlanan dikdörtgen kenarları, simetrik, tekrarlı dalgalar yapan bir geometriye evrilmiştir. Formüllerdeki katsayıların farklı değerlere getirilmesi ile geometrinin

tekrarlı yapısı kırılmış kendi içerisinde daha kompleks bir eğri oluşturulmuştur (Şekil 4.3).



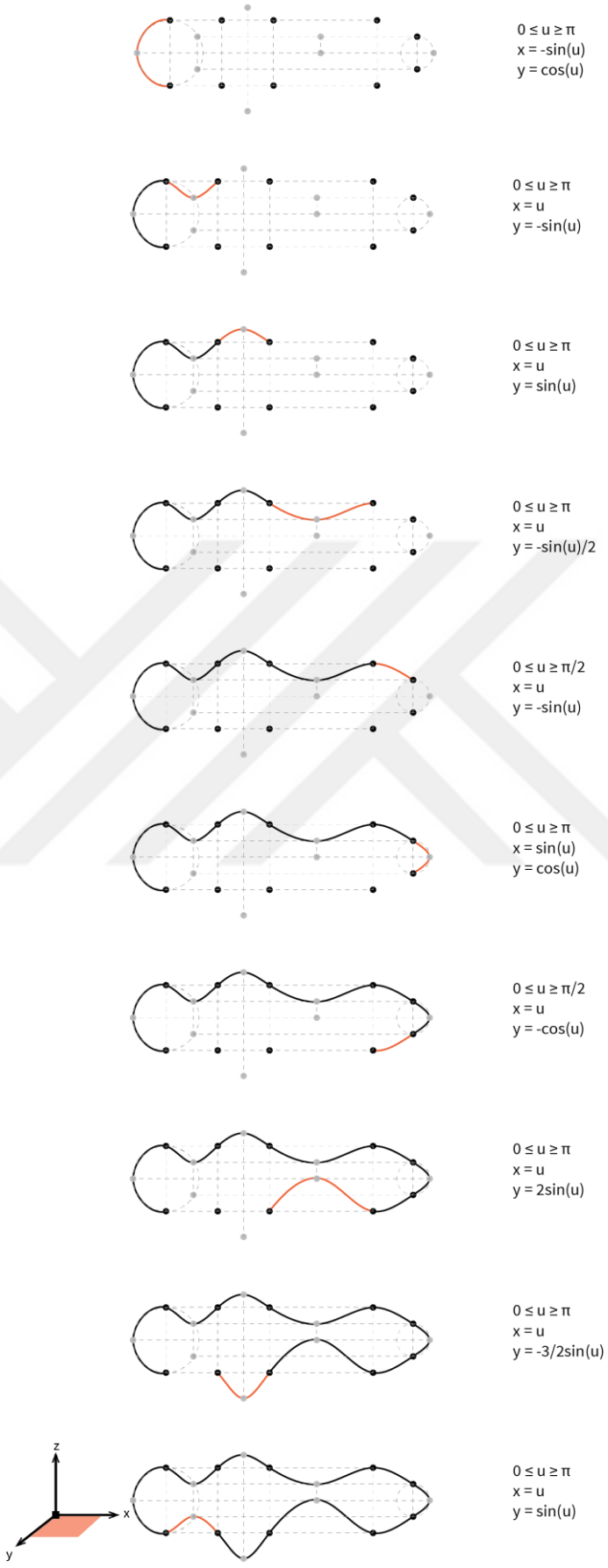
Şekil 4. 3: Konsept Diagramı.

Kapalı eğri tek bir adımda ve tek bir matematik formülü ile oluşturulmamıştır. Geometrinin tamamı, 10 parçaya bölünmüş eğriler kümesinin birleşimi ile meydana gelmiştir (Şekil 4.4).



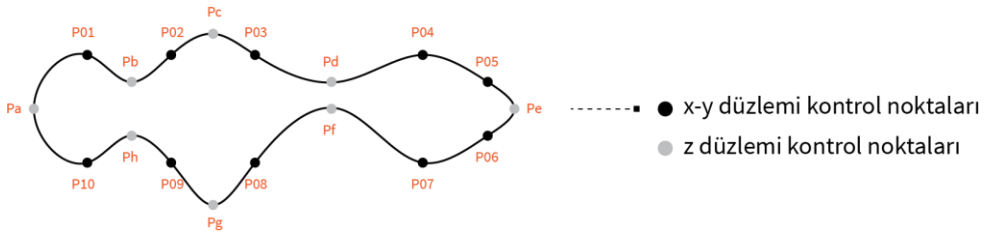
Şekil 4. 4: 2 boyutlu 10 parçalı eğri ve 10 kontrol noktası.

Her bir parça için bir formül kullanılarak, birinci dereceden yazılmış 10 sinüs ve cosinüs denklemi ile, x-y düzleminde kapalı eğriyi oluşturan 10 kontrol noktası oluşturulmuştur (Şekil 4.5). Eğrinin parçalı bir şekilde formülasyonunun yapılma sebebi, formda esnekliği sağlamak adına kontrol nokta sayısı fazla bir geometri elde etmektir. Başka bir ifade ile bu yöntem, ilişki ve katsayıların değiştirilmesi ile daha esnek bir form oluşumuna olanak sağlamıştır.



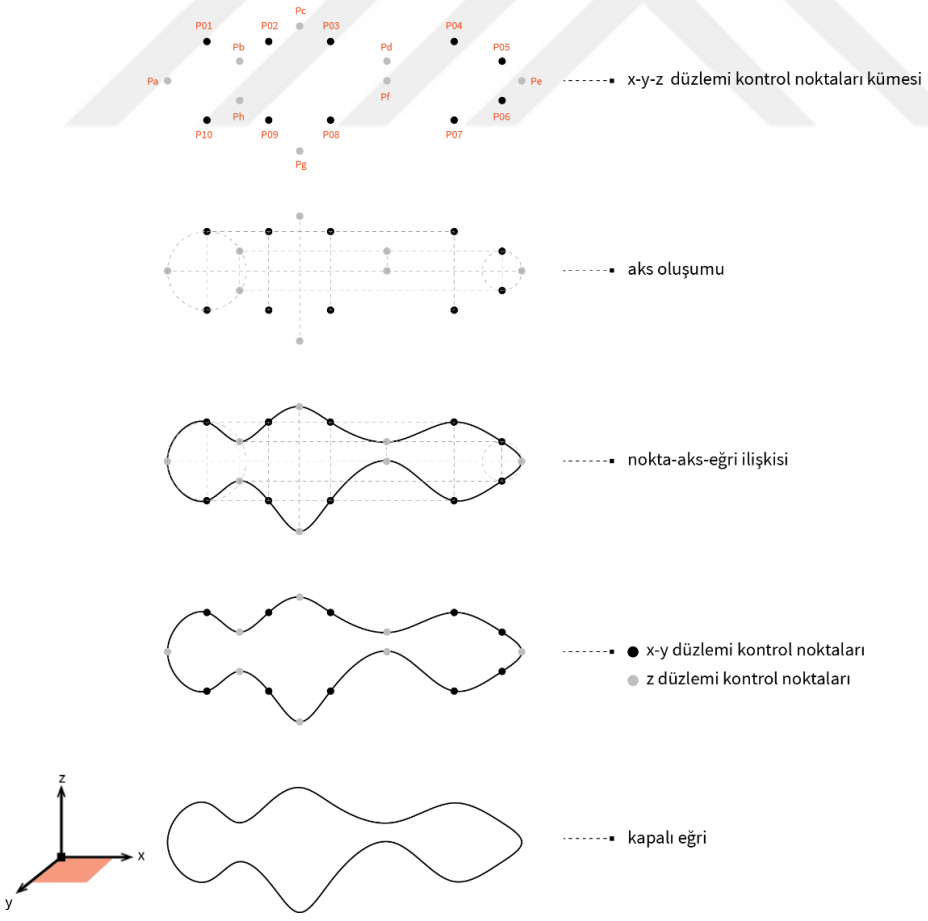
Şekil 4. 5: Formu oluşturan eğrilerin matematiksel formülü.

Geometrinin simetrik, tekrarlı yapısı, başlangıçta her biri aynı katsayıya sahip eğri fomüllerinde katsayıların değiştirilmesi ile bozulmuştur. x-y düzleminde kompleks 2 boyutlu geometri oluşturulduktan sonra, kapalı eğri ile 2 boyutlu yüzey elde edilmiştir. Daha sonraki adımda yüzeyi 3 boyutlu geometriye dönüştürmek için her bir eğrinin orta noktaları belirlenip, kontrol noktaları olarak parametrelere dahil edilmiştir (Şekil 4.6).



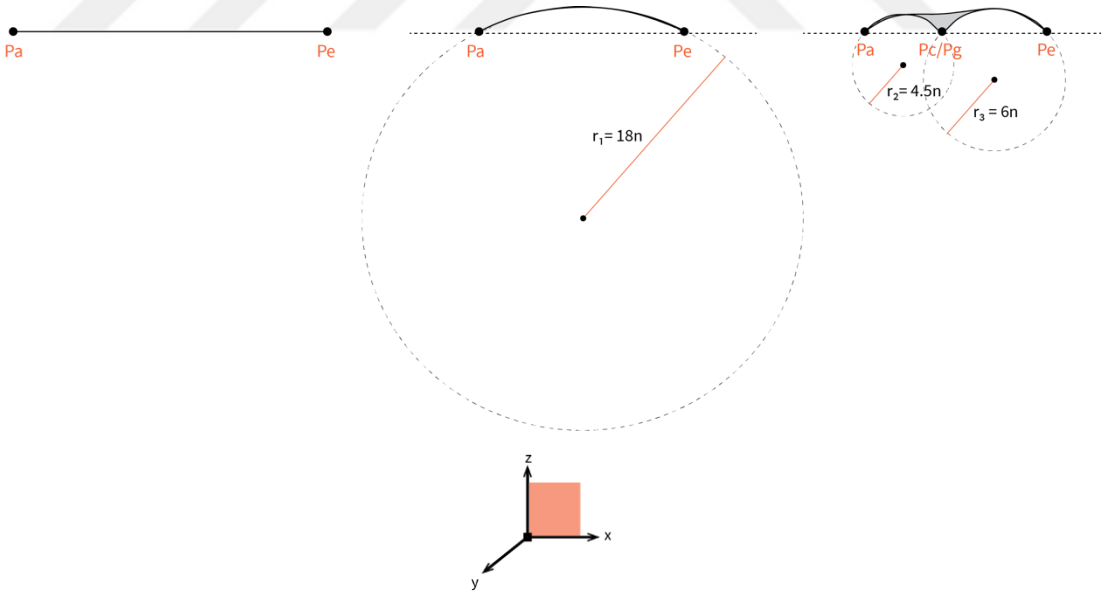
Şekil 4. 6: Geometrinin kontrol noktaları.

Sonuç olarak 18 noktadan kontrol edilebilen, kurallı ve başlangıç ve bitiş noktaları arası akslar birbirleri ile ilişkili, 10 parçalı eğri, 2 boyutlu geometriyi oluşturmuştur (Şekil 4.7).

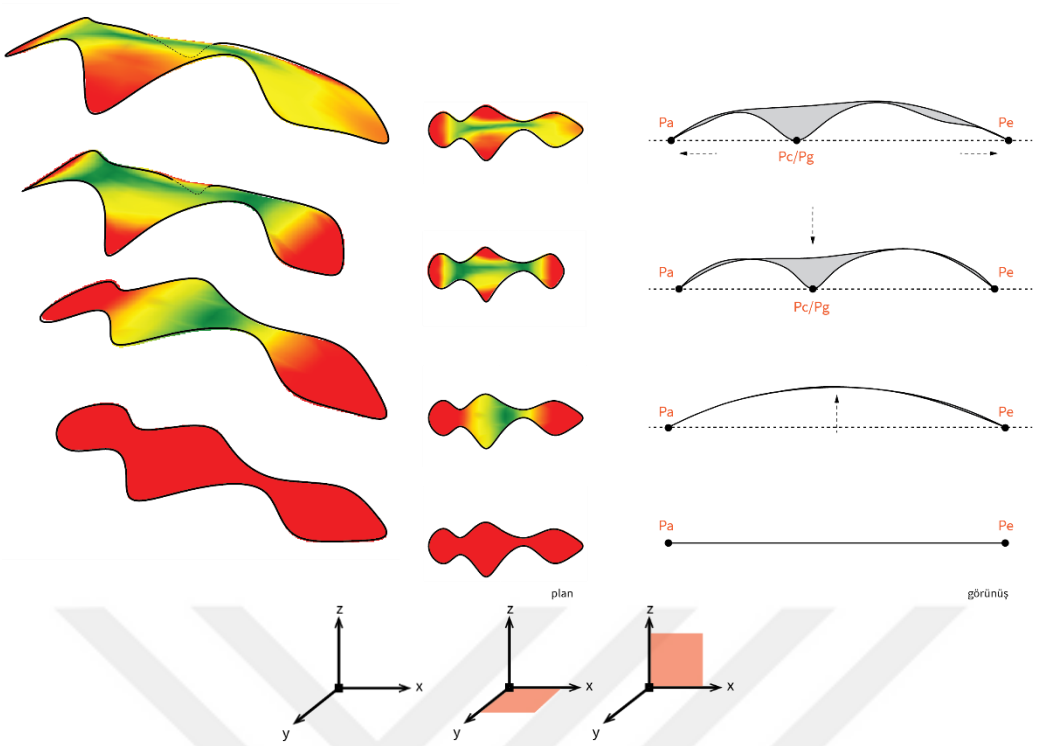


Şekil 4. 7: 2 boyutlu kapalı eğri oluşum şeması.

3 boyutlu geometriye geçiş 3 adımda gerçekleştirilmiştir. z düzlemi kontrol noktaları olan eğrilerin orta noktaları, birinci adımda zemin ile ilişkinin sağlanacağı geometrinin 2 ucu olan Pa ve Pe noktaları yere basacak şekilde yükseltilmiştir. Bu işlemde kurallı bir geometri oluşturmak için formun kesitinde oluşacak eğri bir dairenin yarıçap parametresine (r_1) bağlanmıştır. Pa ve Pe noktalarında z=0 iken diğer noktaların bir dairenin parçası olduğu değerlere ulaşıldığında yarıçapı 18n olan bir daire oluşmuştur. Bu işlemden sonra eğim dağılımını homojenleştirmek için formun orta noktalardan da zemine temas etmesi amacı ile 3 ve 9. cu eğrilerin orta noktaları (Şekil 4.4), Pc ve Pg noktaları (–) yönde hareket ettirilip z=0 konumuna alınmıştır. Bu işlemde de bir önceki adımda olduğu gibi kurallı geometriyi oluşturan daire yarıçapları r_2 ve r_3 değişkenlerine bağlıdır. Pc ve Pg noktaları z=0 noktasına alındığında 4.5n ve 6n olan 2 dairenin parçası olarak geometri oluşturulmuştur (Şekil 4.8). Son adımda kontrol noktaları 2 boyutlu geometrinin iz düşüm noktalarına taşınmış ve eğim dağılımının daha iyileştiği test edilmiştir (Şekil 4.9). Testlerin yenilenerek formun üzerinde daha iyileştirme yapılıp optimum değerlere ulaşmak mümkündür ancak bu çalışmada 3 adımda kalmak yeterli görülmüştür.

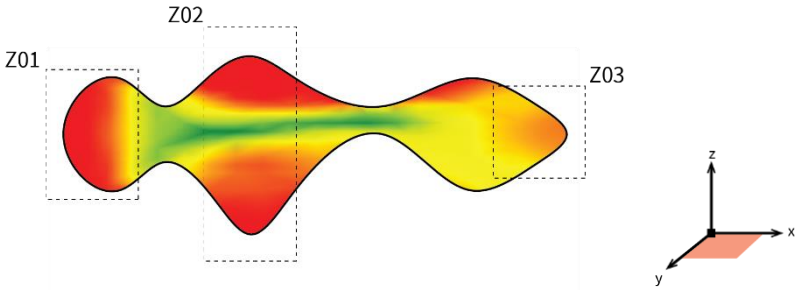


Şekil 4. 8: 2 boyutlu eğriden 3 boyutlu eğri elde etme; z düzleminde yükseltme.

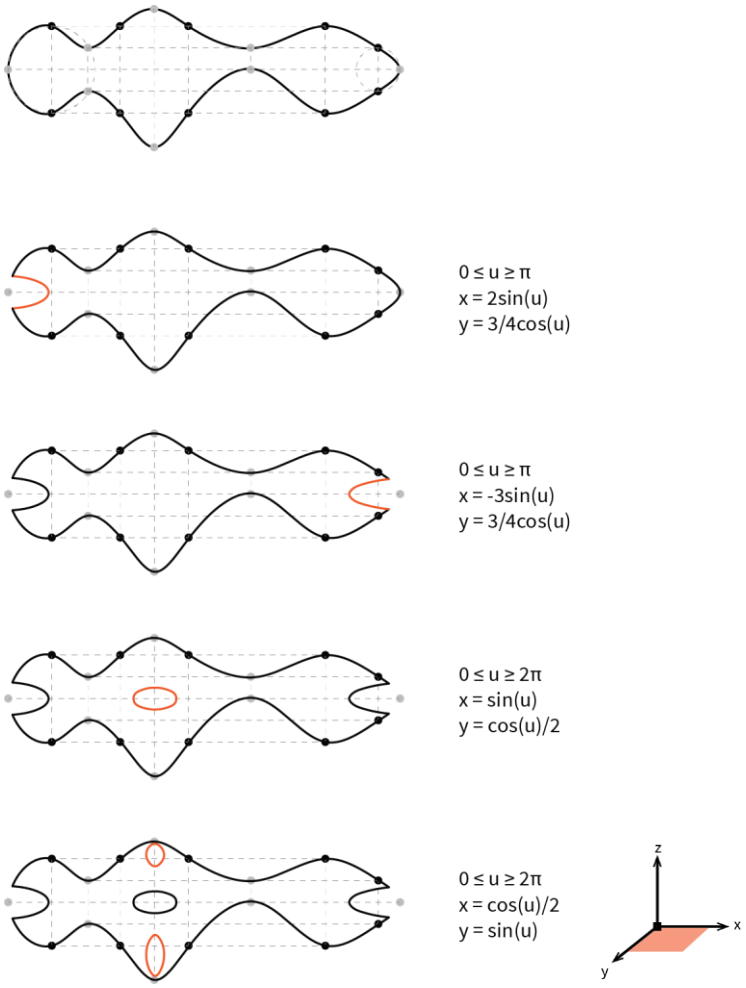


Şekil 4. 9: Eğim diyagramı.

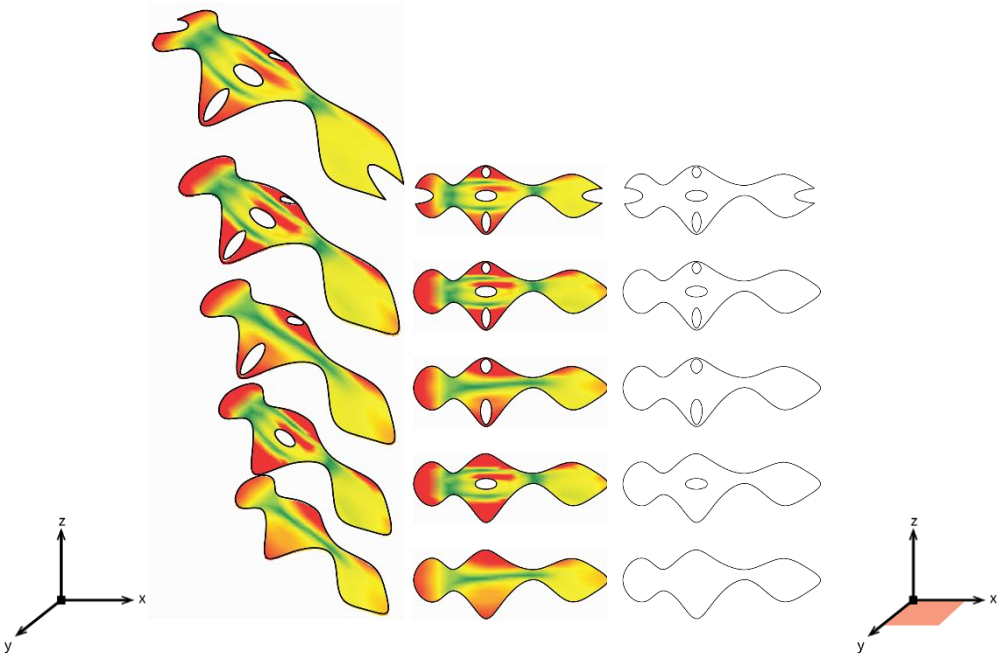
Bir sonraki adımda eğimin fazla olduğu bölgelerde boşaltma işlemi yapılmıştır (Şekil 4.10). Bu adımda amaç, kabuk yüzeyinin ve kabuğu örten panellerin kabuk yüzey eğimi ve açıklıklara göre optimize edilme sürecini çalışabilmek adına alan oluşturmaktır. Bu adımda da çizilecek geometriler için matematik formülleri kullanılmıştır (Şekil 4.11). Her açıklık oluşturulduktan sonra adım adım analiz süreçleri devam etmiştir (Şekil 4.12).



Şekil 4. 10: Eğimin fazla olduğu bölgeler.



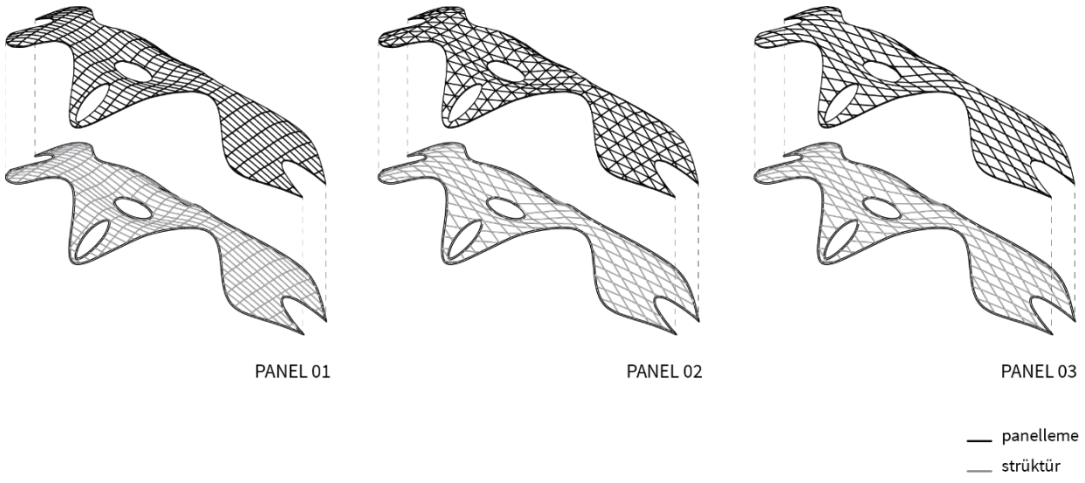
Şekil 4. 11: Açıklıkları oluşturan eğrilerin matematiksel formülü.



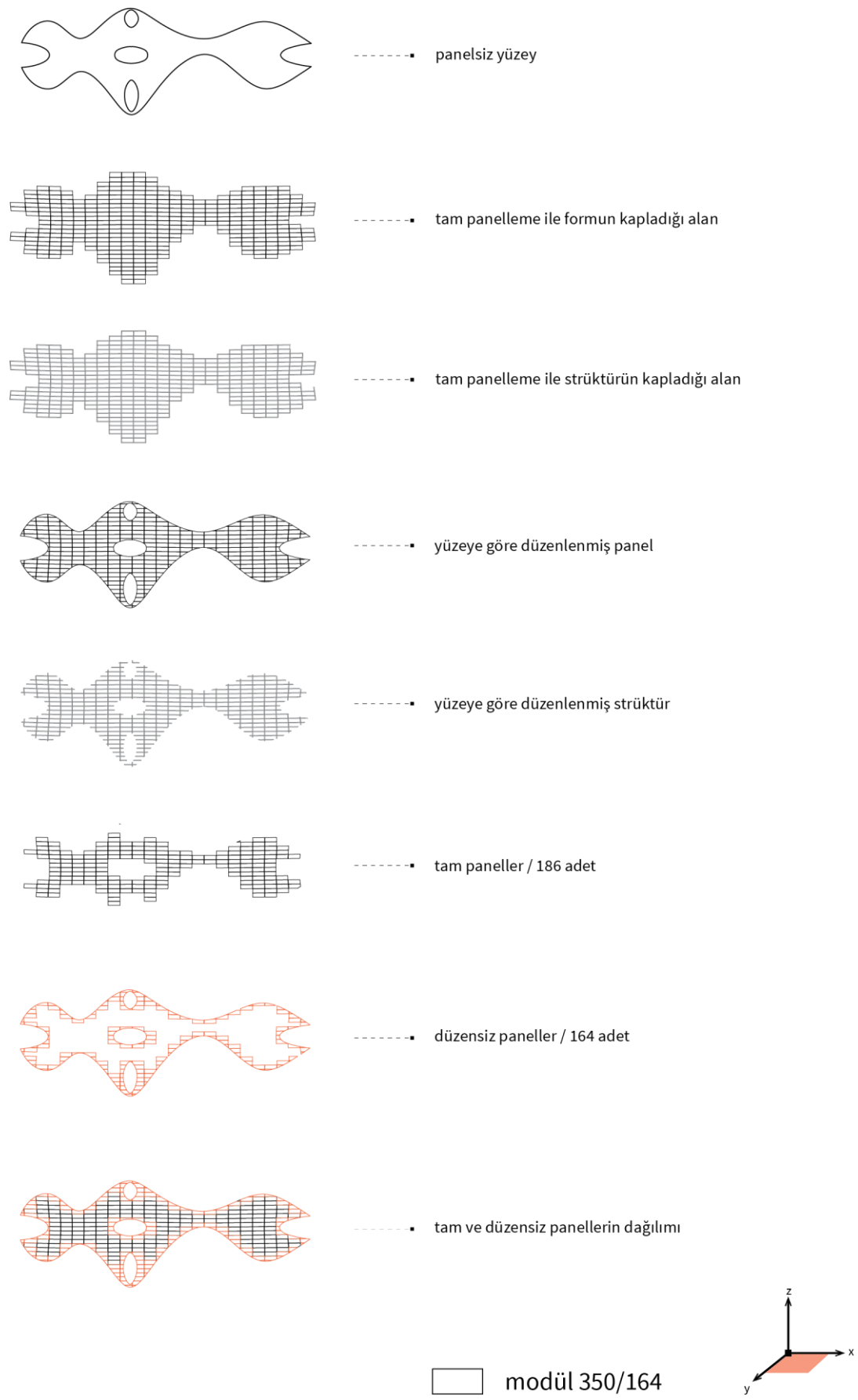
Şekil 4. 12: Eğim analizine göre açıklıkların oluşturulması.

Formun geometrisinin oluşum adımlarında olduğu gibi açıklıklar için de analizler devam ettirilip en iyileştirme işlemleri geometri üzerinde sürdürülebilir. Bu noktada çalışma kapsamında form üzerine uygulamalar dondurulup panelleme işlemlerine geçilmiştir.

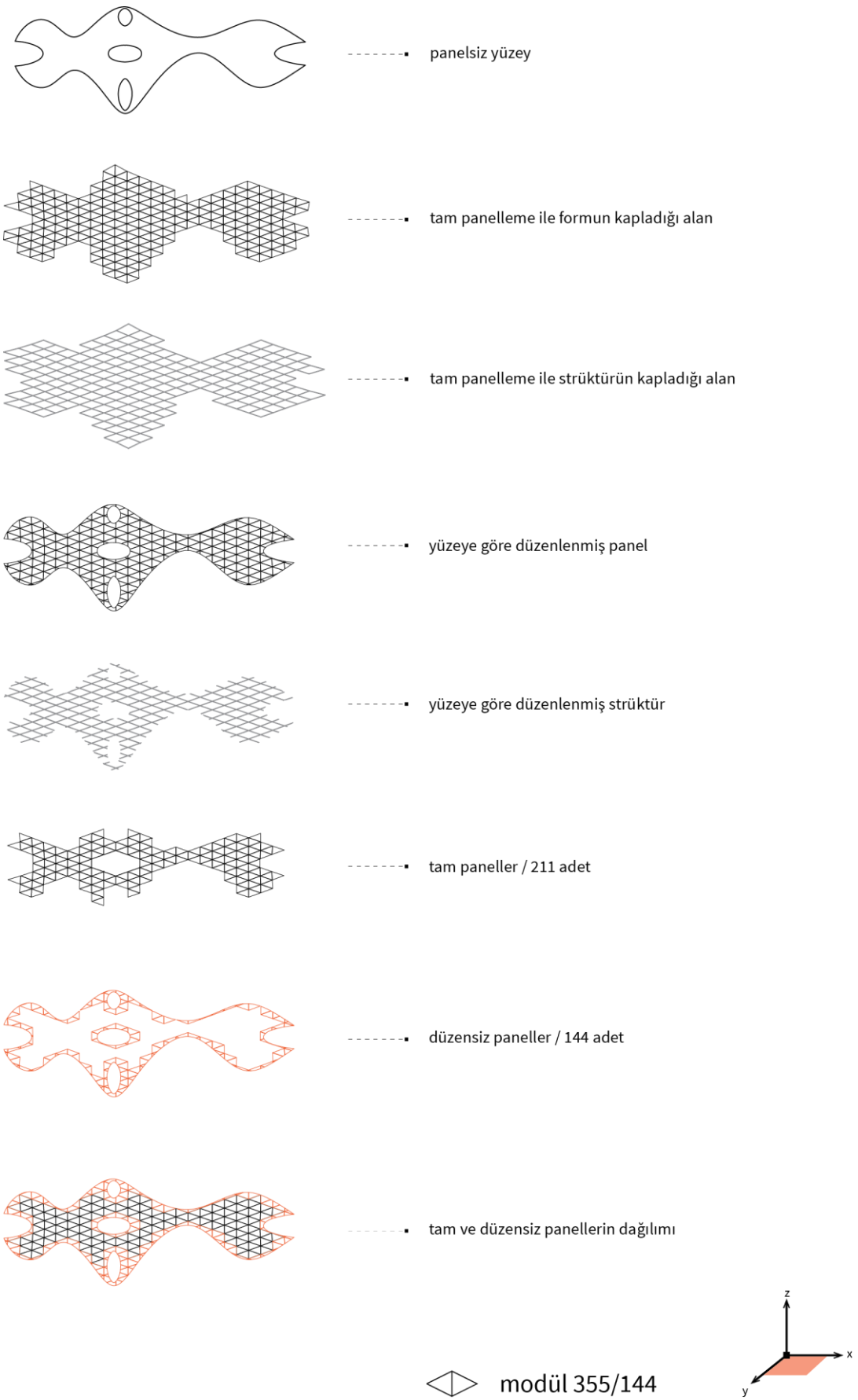
Panelleme adımında da mantıksal bir kurgu oluşturulmuştur. Alt strüktür ve panel boyutları ile panel modülünün formu üzerine 3 adımda denemeler yapılmıştır. Her bir adımda panel boyutları ve modül geometrisi, panel sayısı ve aynı modülün tekrar sayısı göz önünde bulundurularak, bu sayıların en iyileştirilmesi için değiştirilmiştir (Şekil 4.13). Başka bir ifade ile amaç, minimum sayıda panel sayısı içinde yine minimum sayıda düzensiz panel maksimum sayıda tam panel elde etmektir. İlk adımda 70x25 cm ölçülerinde dikdörtgen modül kullanılmıştır. Buna paralel olarak alt strüktür de dikdörtgen panel modülüne uygun seçilmiştir. Bu panellemede 350 adet panel kullanılmış, 186 tam modül, 164 düzensiz modül oluşmuştur (Şekil 4.14). İkinci panelleme adımında modül geometrisi, taban kenar uzunluğu 50 cm yüksekliği 70 cm ölçülerinde ikizkenar üçgen ile oluşturulmuştur. Alt strüktür üçgen ızgara olacak şekilde yapılmıştır. Bu kez panel sayısı 355 olmuş, bunlardan 211'i tam, 144'ü düzensiz modülden oluşmuştur (Şekil 4.15). Buradan formun eğrisel geometrisinin panellemede üçgen modül ile daha iyi sonuç verdiği gözlenmiştir. Bu çıkarımdan hareketle son adımda panel modülünde iki ikizkenar üçgen sırt sırta birleştirilip baklava desenine geçilmiş, alt strüktürde değişim yapılmamıştır. Bu sayede modül sayısı hemen hemen yarıya inerek, 193 olmuş, bunlardan 101 adedi tam, 92 adedi düzensiz modül olmak üzere panelleme yapılmıştır (Şekil 4.16).



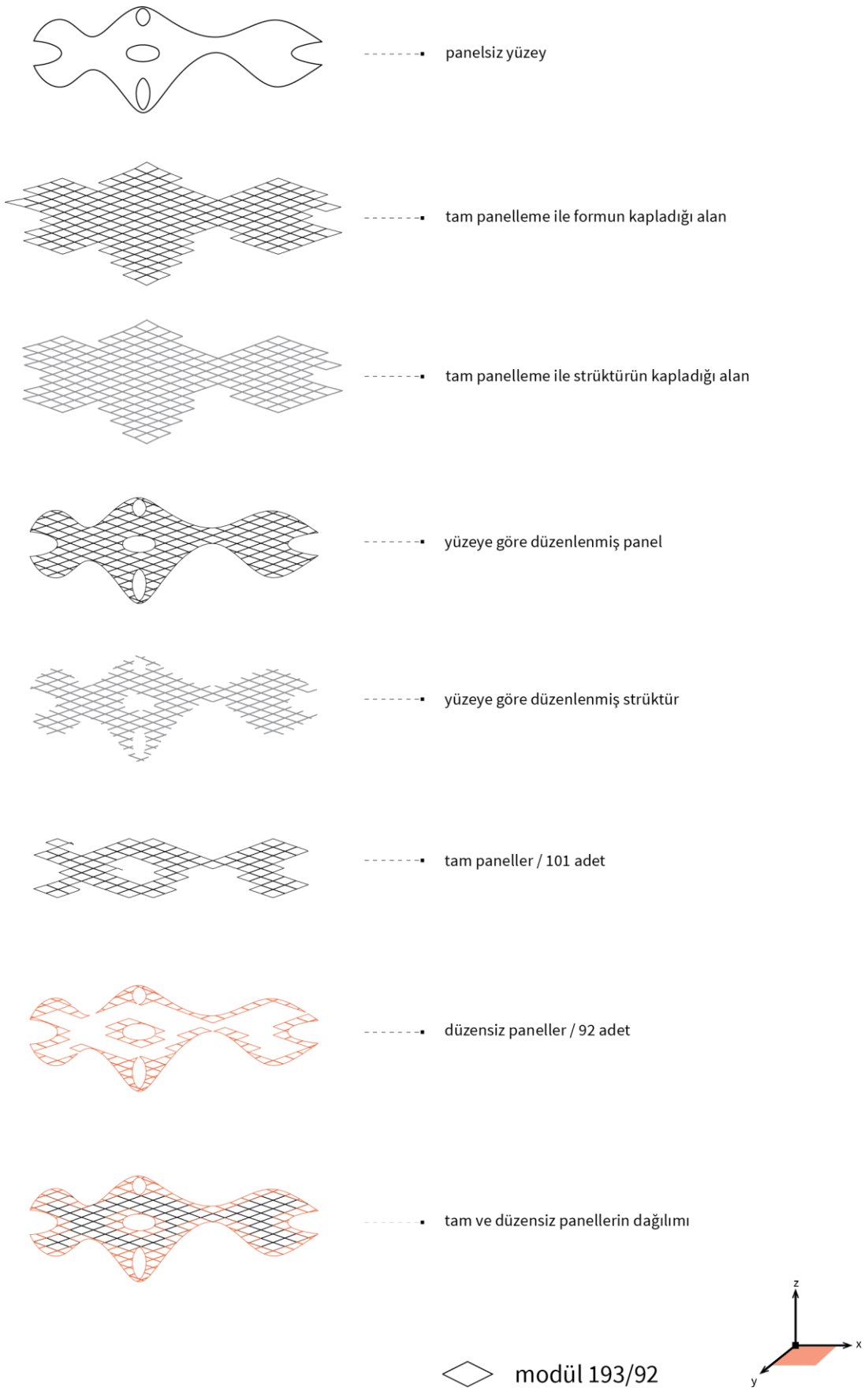
Şekil 4. 13: Panel varyasyonları.



Şekil 4. 14: Sayı ve çeşitleri optimize ederek dikdörtgen panelleme.



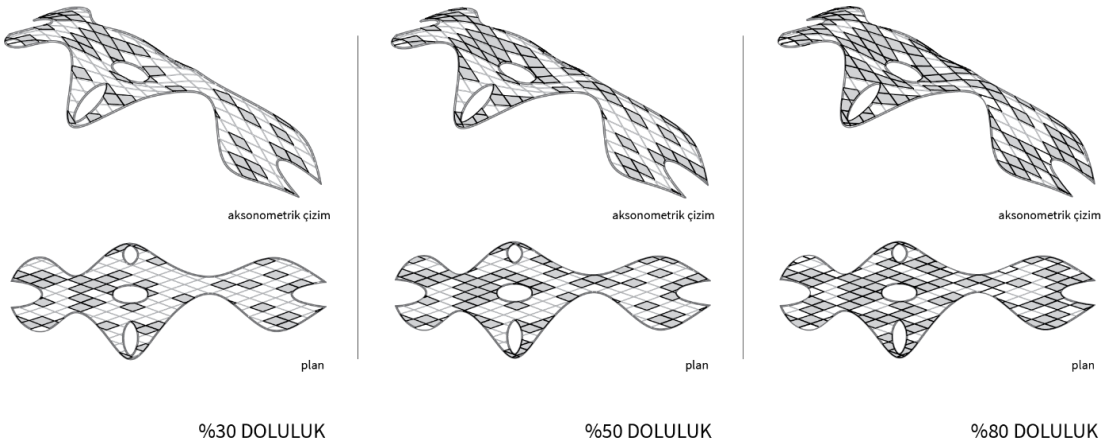
Şekil 4. 15: Sayı ve çeşitleri optimize ederek üçgen panelleme.



Şekil 4. 16: Sayı ve çeşitleri optimize ederek baklava panelleme.

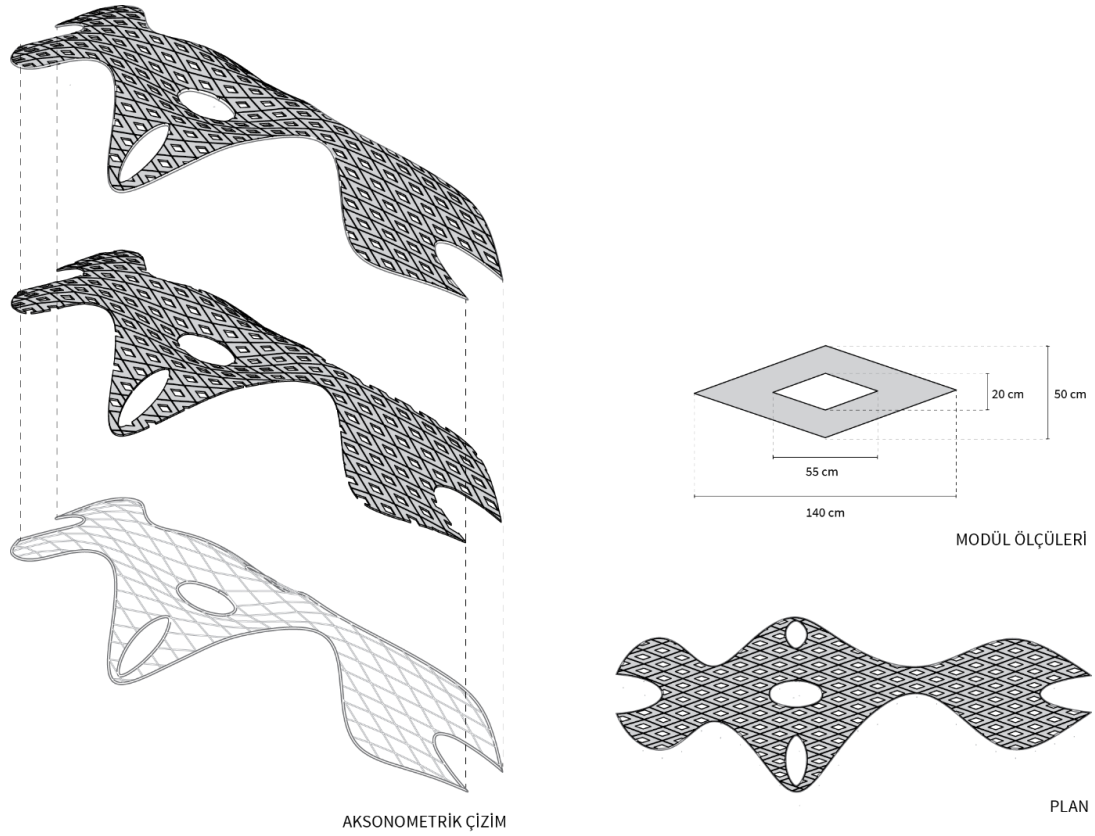
Formda oluşturulan açıklıklar düzensiz panel sayılarını arttırmaktadır. Burada izlenen yöntemde, açıklıkların geometrisi sabit tutulup, panel modüllerinde değişikliğe gidilmiştir. Ancak geometri bir algoritma üzerinden oluşturulduğu ve değerler değişkenler olduğu için, bir diğer senaryo, panel modülünün sabit düşünülüp, düzensiz modül sayısını azaltmak için açıklıkların formunda değişikliğe gidilmesi olabilir.

Bir sonraki adımda panellerde de eksiltme işlemi uygulanmıştır. İlk adımda kuralsız bir şekilde rasgele seçim yapılmış gridlerde paneller eksiltilmiştir. Yapılan 3 denemede %30, %50 ve %80 oranlarında doluluk sağlayacak şekilde eksiltme işlemi yapılmıştır (Şekil 4.17).



Şekil 4. 17: Panellerde rastgele seçim ile eksiltme.

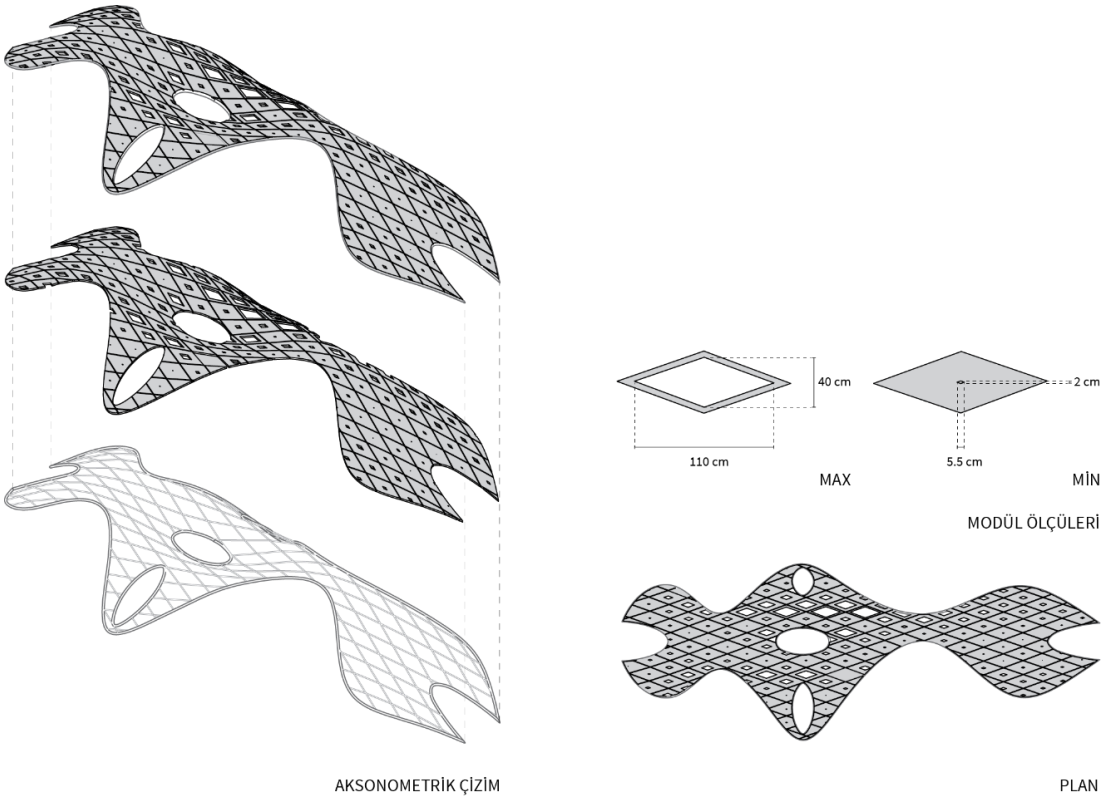
Daha sonra eksiltme işlemi yerine panellerde açıklık oluşturma denemesi yapılmıştır. İlk adımda düzenli bir şekilde panellerin tamamına aynı ölçülerde açıklıklar uygulanmıştır (Şekil 4.18).



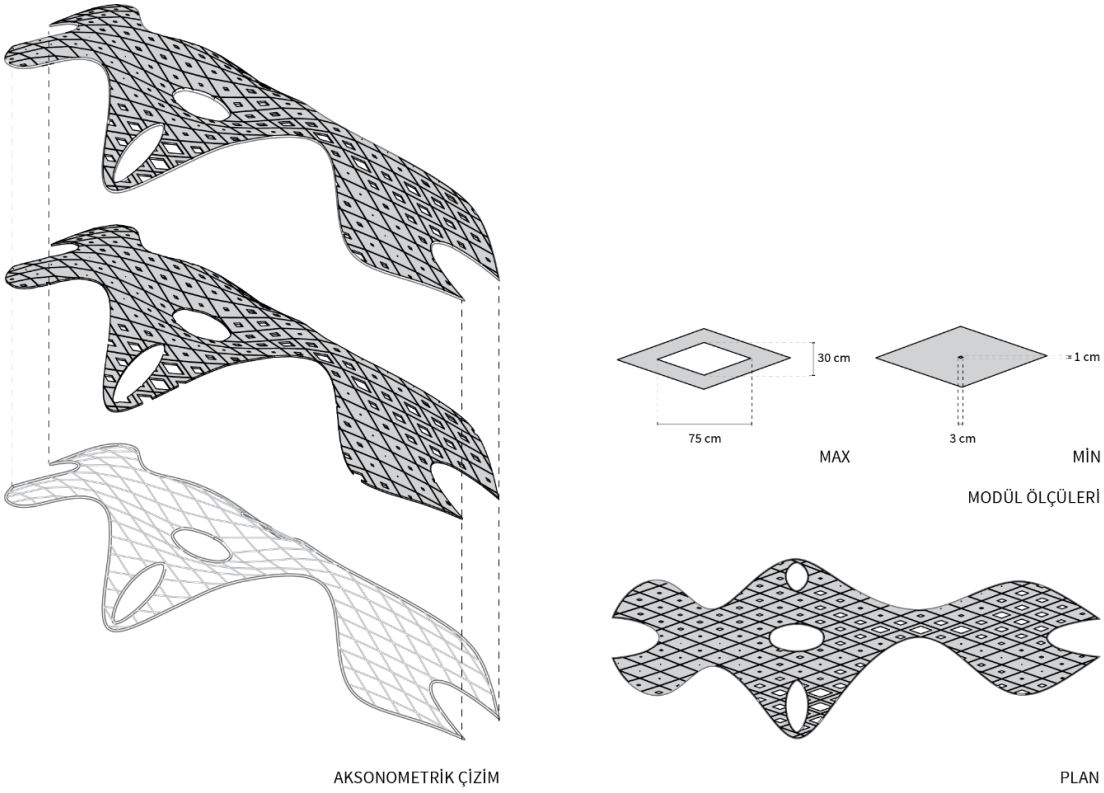
Şekil 4. 18: Panellerde açıklık oluşturma.

Son olarak panellere uygulanan açıklıkların eğim değerlerine göre değişen oranlarda açıklık oluşturma denemesi yapılmıştır.Burada formun yüzey eğim değerleri hesaplanıp ilk adımda eğimin fazla olduğu bölgelerden az olduğu bölgelere doğru gittikçe büyüyen açıklıklar için bir algoritma geliştirilmiştir, başka bir deyişle eğimin fazla olduğu bölgeler küçük açıklıklara sahiptir (Şekil 4.19). İşlemin ikinci adımında ise değerler tam tersi işlemle, eğimin fazla olduğu bölgelerden az olduğu bölgelere doğru gittikçe küçülen açıklıklar oluşturacak şekilde, eğimin fazla olduğu bölgelerde büyük açıklıklar ile değiştirilmiştir (Şekil 4.20).

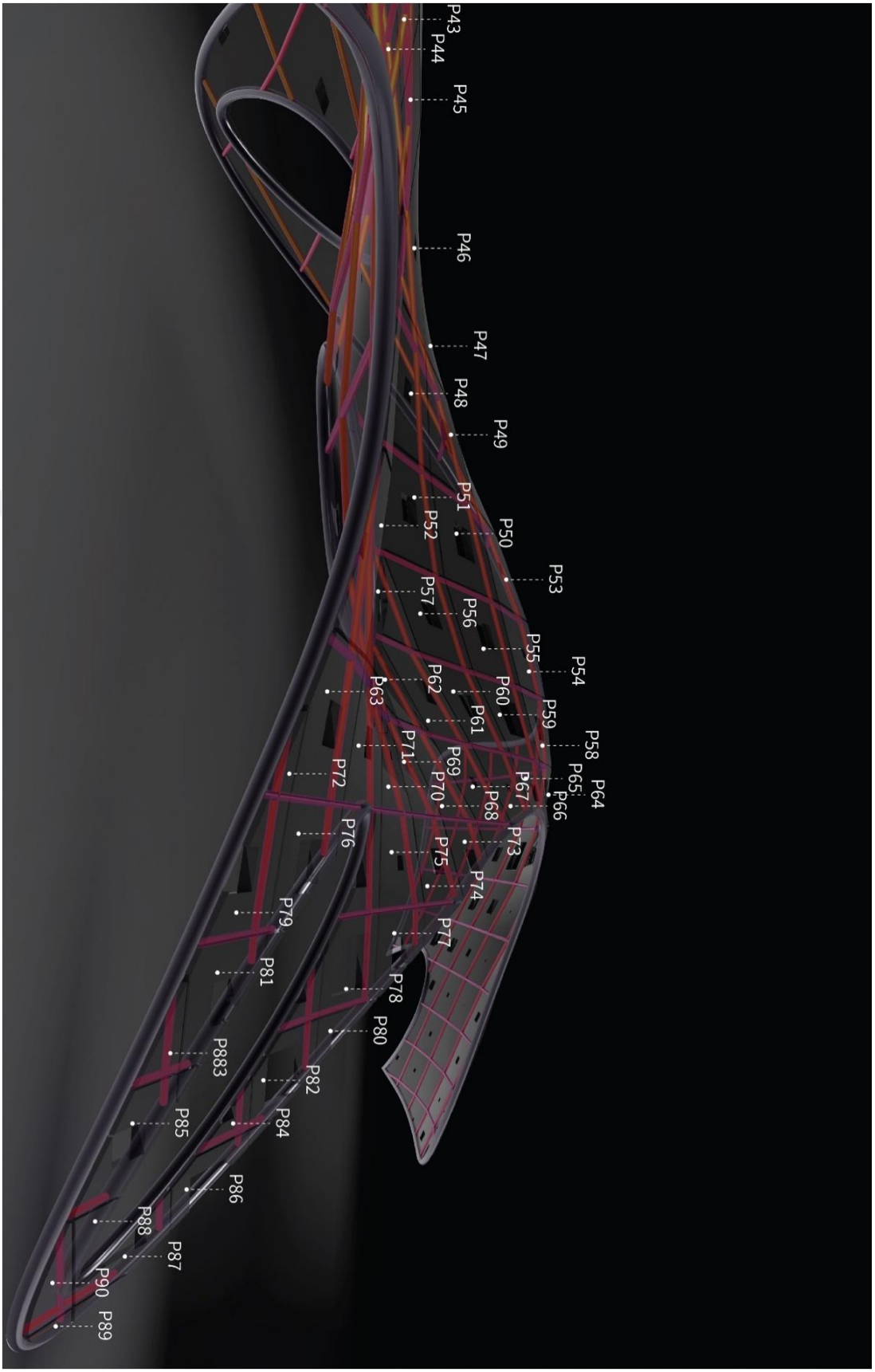
Algoritma yardımı ile optimize edilerek oluşturulmuş tüm olası formalar arasında, nihai forma olarak bir olasılık seçilmiş, süreçte baştan sona bu nihai form hedefi ve fikri olmamak ile birlikte sonuç, süreç içerisinde bulunan olmuştur (Şekil 4.21 - Şekil 4.22 - Şekil 4.23).



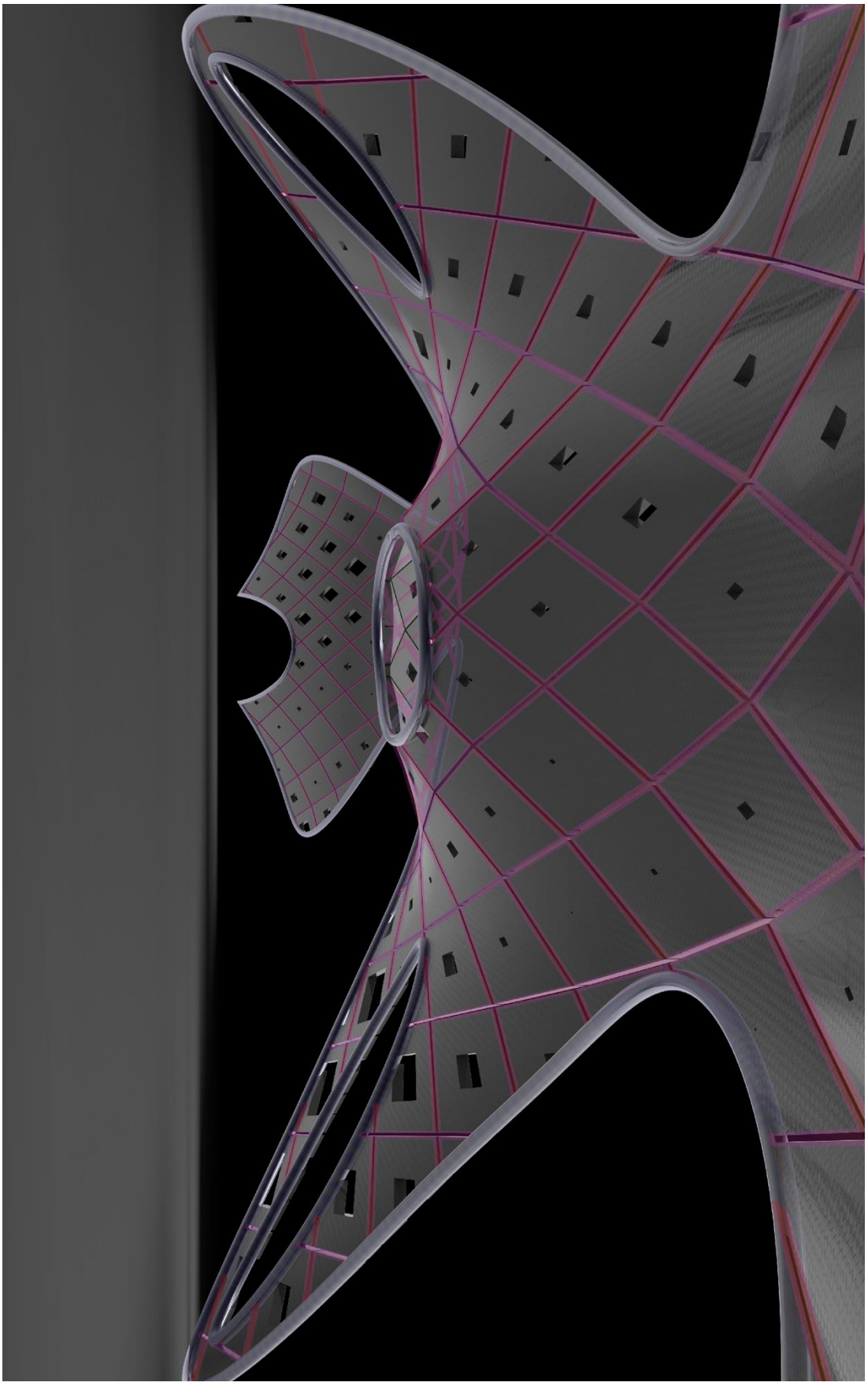
Şekil 4. 19: Eğim değerlerine göre panellerde açıklık oluşturma, varyasyon 01.



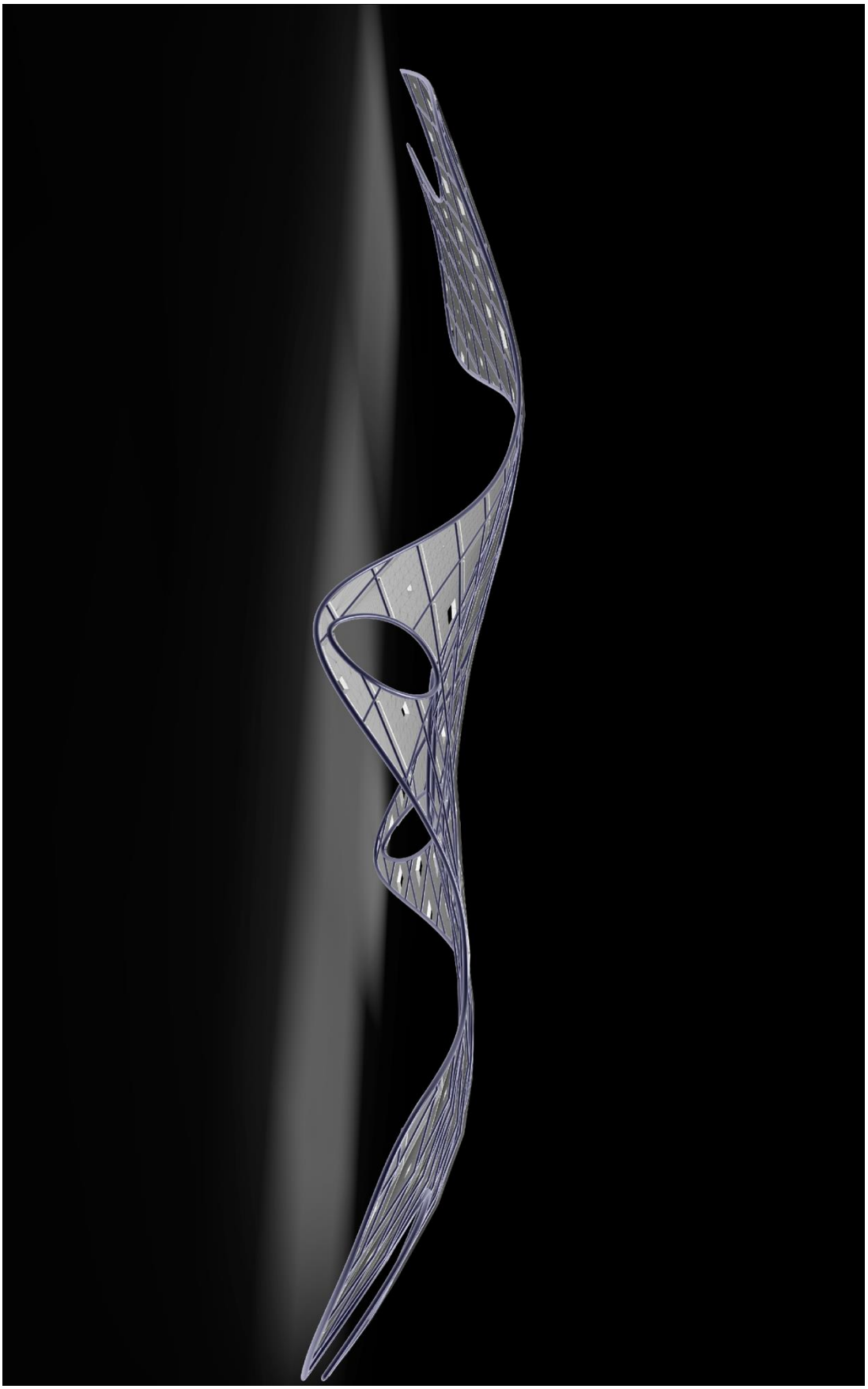
Şekil 4. 20: Eğim değerlerine göre panellerde açıklık oluşturma, varyasyon 02.



Şekil 4.21: Görsel 01.

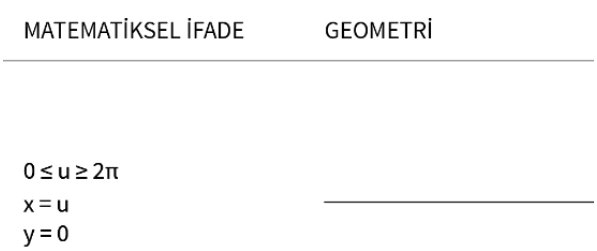


Şekil 4.22: Görsel 02.



Şekil 4.23: Görsel 03.

Tüm bu adımlar arası geçişlerde aslen birbirini besleyen bir süreç akışı görülmektedir. Bir önceki denemede öğrenilenler bir sonraki adımın hedefini oluşturmuştur. Form denemelerinin oluşumunda matematiğin iki seviyede etkisi ele alınmaya çalışılmıştır. Birinci seviyede ana (primer) etki gözlenir ki doğrudan doğruya geometrinin oluşturulmasında formüller kullanılmıştır. İkinci seviyede ise ikincil (seconder) bir etkisinden söz etmek mümkündür. Bu seviyede geometrinin oluşumunda ilişkiler kurabilmek için matematik tek araç olarak görülmüştür. Aslen birinci seviyede de bir tanımlama geometrinin kendisinin oluşumunda ilişkiler tanımı söz konusudur. Örneğin çizginin tanımı, u değerleri 0 ile 2Pi arasında iken, y'nin bu değer aralığında 0, x'in u değerine eşit olması ile yapılır. Çizgi oluşumunu sağlayan x ve y şartları vardır. Kendi içerisinde bu ilişki tanımını yapan formül geometrinin kendisini birinci dereceden oluşturmaktadır (Şekil 4.24).



Şekil 4. 24: Çizginin matematiksel ifadesi (L_1).

Ancak oluşturulmuş L_1 çizgisi ile ilişkili ikinci bir L_2 çizgisinin tanımında ikinci seviyede bir matematiksel tanımlama yapılır. Yeni oluşturacak çizginin başlangıç noktasının L_1 çizgisi orta noktası ve bitiş çizgisinin L_1 çizgisinin orta noktasından +y yönünde 5n uzakta olacağı tanımı yine bir çizgi tarifi yapmakla birlikte geometriyi oluşturmada gerekli ilişkilere değil, L_1 çizgisi ile ilişkiye bağlı olarak yapılmaktadır. L_1 çizgisi temel geometri tanımı ile oluşurken, L_2 çizgisi matematiğin ikinci seviyesi ile ilişkilendirilmiş geometri olarak tanımlanır.

Kabuk tasarımında ikinci boyutta oluşturulmuş kapalı eğri formunda temel geometri oluşum algoritmalarını daha baskın bir şekilde gözlemek mümkünken, hesaplama ve analizler doğrultusunda oluşturulan formlarda, panelleme denemelerinde, hesap ve ilişki tanımları ile ilişkili geometri oluşumunu gözlemlemek mümkündür.

Sonuç olarak, üretilmiş pek çok deneme, formun kendisinin sürekli belirip kaybolduğu bir olasılıklar aralığında, birer olasılıktır.

5. SONUÇ

“Matematik mimarın gücünü arttırırken, ona makul ölçüde amaçlayabileceği sınırları da hatırlattı” (Picon, 2011, s.30).

Tasarım bir süreçtir. Soyut ve dışlaştırılması güç olarak görülen tasarım süreci, yalnızca sonucun değerlendirilmesi ile sınırlara tabi tutulurken, işlemsel tasarım, bütünüyle tasarım fiilini ele alarak sürecin deşifrelenmesine olanak sağlamıştır. Süreç içerisinde yaklaşımlar, fikirler, öngörüler, aslen tüm soyutluklar, somut birer veriye dönüşmeye başlamıştır. Bunun işlemsel tasarımın matematiksel alt yapısı ile mümkünleştiği yaklaşımı, matematiksel düşünce ve matematiksel dil sayesinde tasarım ile iletişim kurulduğu ispatlanarak gösterilmiştir. Başka bir ifade ile, yalnızca tasarım çıktısı değil, tasarım girdileri de işlemsel tasarımda somutlaştırılıp tasarım problemi bir sistem içerisinde ele alınabilir olmuştur.

İşlemsel tasarım bir düşünce sistemidir. Teknoloji ile gelinen bir nokta değil, insanlık gelişimi boyunca takibini yapabileceğimiz, insanda ve doğada mevcut işleyişin içerisinde okumanın mümkün olduğu bir sistemdir. Bu teknolojinin bugünkü boyutlarına ulaşmadığı dönemlerde gerek tasarım alanında gerekse diğer disiplinler içerisinde keşfi yapılan yaklaşımlardaki akıl ile gösterilmiştir. İnsan, sistematik düşünce yapısını somutlaştırdıkça gelişmeler ve değişmeler beraberinde gelmiş, bu akla hizmet eden araçlar, teknoloji, olarak ortaya çıkmıştır. Soyut ifadeler, teknolojinin sunduğu imkanlar ile somut verilere dönüştürülebilir ve kontrol edilebilir olmuştur. Bugün geldiğimiz noktada, dünyada 1 yıl içerisinde 16.3 zettabyte veri üretebilir hale gelmiş durumdayız ve bu rakam dramatik bir hızla büyümeye devam etmektedir (Url-33). Ancak %80 ini kullanamadığımız bu veri madeninin nasıl kullanılabileceği, neler sağladığı, yeni çağda ortaya çıkan yeni iş modelleri tartıştığımız konular olmuştur. Dünya üzerinde herşey sayısallaşıp, sistemler ile kontrol edilmeye çalışılırken, tasarım mecrasında işlemsel tasarım, yeni çağa kayıtsızlığın önüne geçmemizin fırsatlarını sunmaktadır.

İşlemsel tasarım, tasarım eylemi ile, formun kendisi ile matematiksel bir iletişimin temeline dayanan bir düşünce sistemidir. Bu matematiksel alt yapı sayesinde form ile

kurallar dahilinde iletişim kurulmaya, form hesaplanmaya ve çizilmenin de ötesinde üretilmeye başlanmıştır. Sistem içerisinde formun matematiksel düzenleme ile nasıl oluşturulduğu ve kontrol edildiğinin anlaşılması vardır, başka bir ifade ile "işlemsel tasarım, formu farketmekten, anlamaktan formun direkt olarak nasıl üretildiğinin çözümlenmesine geçiş sağlamıştır" (Menges ve Ahlquist, 2011, s.10). Tasarlama eyleminin yeni bir tarifi oluşmuş ve aslen bir üründen bir zaman tasarımına, bir sonuçtan birden çok olasılık üreten bir sistem tasarımına, tekilden çoğula ve bir çözümden, bir akıl üretimine geçilmiştir. Belirsizlik bulutu içerisinde görünen bu tarif, en basit şekli ile tasarımda nihailikten sıyrılmayı anlatmaktadır. Bir tasarım probleminin birden çok çözümü, birden çok girdisi, birden çok etkeni mevcut iken, tasarım sürecinin de bu esnek yapıya uyum sağlaması tariflenmektedir. Tasarımın bir elemanı olan ve halen araç olarak ifade ettiğimiz dijital dünya ise araç olmanın ötesinde, tasarımcıya yön veren bir platforma dönüşmüş, araçların ve araçlara olan hakimiyetin de önemi artmıştır. Tüm bu nihaisizlik içerisinde, matematiğin kesinliği, aslen ekranda belirmemiş ancak bir okadar da net kuralları olan formu oluşturur. İşlemsel tasarımda ilişki ve değerleri kurallandırmak, ilişkilendirmek, değişebilir kılmak bir fantazi değil gerçeklikle, madde dünyası ile dijital ortamda karşılaşmadır. İşlemsel tasarımın aklı matematiksel bir dil ile uygulanabilir bir gerçeklik oluşturmaktadır. Sonuçta değişen tüm eylemler ve kavramlar tasarımcının düşünme ve tasarlamasını değiştirmiştir (Lynn, 2006).

Mimarlık işlemsel tasarım ile, yalnızca sabit form ve kataloglardan kurtulmamış, aynı zamanda alanda yeni çağın iş modeline de ayak uydurmanın önü açılmıştır. Tasarımın üretilebilirlik bilgisi, fikirlerin sayısallaşması, sistematize etmenin yapabilme sınırlarını çizmesi işlemsel tasarım ile konuşmaya başladığımız konular halini almıştır. Süreç tasarımının başlı başına alanda bir platforma dönüşmesi, eskiden somutlaştıramadığımız için bilemediğimiz tasarımcı fikirlerinin, bugün bilgiyi koruma ve saklama sebebi ile bilemediğimz kapalı tasarım sürecine dönüşmesi, alanın öneminin her geçen gün daha da anlaşılır olmaya başladığını göstermektedir.

Bugün halen işlemsel tasarım ayrı bir uzmanlık alanı olup, tasarımcılar, mühendisler arasında alanda uzmanlaşan, kişilerin hayata geçirdiği bir sistemdir. Ancak gelecekte düşünce sisteminin tabana yayılması, mesleki eğitim ile tasarım öğretisi olması sayesinde, ayrılaşma düşüncesi yok olup, herkesin aynı akıl ve dil üzerinde konuşabileceği temel bir sistem olması öngörüsünü yapmak çok da zor değildir.

Bilgisayar kullanımı, dünün bugünkü rüyası iken; bugünün fikri işlemsel düşünce de yarının gerçeği olacaktır (Wing, 2006, s.33).

KAYNAKLAR

Agkathidis, A., (2015). Generative Design: Form-Finding Techniques in Architecture, London, Laurence King Publishing

Akcay, A., (2013). *Programlama Dilleri* [Slides], Erişim <https://www.slideshare.net/arifakcay/programlama-dilleri-16164605>

Alberti, L., B., (1755). The Architecture of Leon Battista Alberti In Ten Books, London, Edward Owen, Erişim <http://archimedes.mpiwg-berlin.mpg.de/>

Anay, H., ve Özten, Ü., (2011). Biçim ve İşlev, Eskişehir, Eskişehir Osmangazi Üniversitesi Yayınları

Aranda, B., & Lasch, C., (2007). Pamphlet Architecture 27 / Tooling, New York, Princeton Architectural Press

Aspray, W., (1990). Logic Machines, W. Aspray (Edt.), Computing Before Computers, 99-121. Iowa, Iowa State University Press

Ayten, U., E., (2010). *Algoritma ve Programlama* [Slides] Erişim http://www.yildiz.edu.tr/~ayten/algortimaveprogramlama_bolum1-2.pdf

Ballantyne, A. (2010). Mimarlar için Deleuze ve Guattari, Yapı-Endüstri Merkezi Yayınları, İstanbul.

Barrow, J., D., (2002). The Constants of Nature From Alpha to Omega the Numbers That Encode the Deepest Secrets of the Universe, New York, Pantheon Books

Benian, E., (2011). Mimar Sinan ve Osmanlı Cami Mimarisinin Gelişimindeki Rolü, Erişim<http://www.vizyon21yy.com/documan/Egitim_Ogretim/Onemli_Insanlari/Turk_Bil_Insnl/Mimar_Sinan/Mimar_Sinan_ve_Osmanli_Cami_Mimarisinin_Gelisimindeki_Rolu.pdf>

Berloquin, B., (2008). Hidden Codes Grand Designs Secret Languages from Ancient Times to Modern Day, New York, Sterling Publishing, Erişim <https://books.google.com.tr/books?id=F9q8BAsXTWEC&pg=PA117&lpg=PA117&dq=vitruvius+man+computational+design&source=bl&ots=fCgiMTC93R&sig=GoYtJPnww7gBZE70f8JGWh1oKUA&hl=tr&sa=X&ved=0ahUKEwiHpaDz0erTAhWlLsAKHYatCI44ChDoAQggMAA#v=onepage&q=vitruvius%20man%20computational%20design&f=false>

Beşlioğlu, B., (2011). An Inquiry Into The Computational Design Culture In Turkey: A Reinterpretation Of The Generative Works Of Sedad Hakkı Eldem and İlhan Koman, Intercultural Understanding, (Vol. 1), 9-15.

Bilgin, H., (2006). Mimar Sinan Yapılarında Kubbeli Örtü Sistemlerinin Yapısal Analizi, S. Ü. Müh. – Mim. Fak. Derg., (Vol 21), 119-127.

Borgmann, A., (2011). Intelligence and The Limits of Codes, T. Bartcherer & R. Coover (Edt.), Switching Codes: Thinking Through Digital Technology In The Humanities and The Arts, 184-188. Chicago and London, The University of Chicago Press

Bosia, D., (2011). Long Form and Algorithm, Mathematics of Space, Architectural Design, 81/4, London, John Wiley

Bromley, A. G., (1990). Difference and Analytical Engines, W. Aspray (Edt.), Computing Before Computers, 3-58, Iowa, Iowa State University Press

Bundy, A., (2007). Computational Thinking is Pervasive, J. Scient. Pract. Comput. (1), 67-69, <https://pdfs.semanticscholar.org/d3b5/562aa8399ecbdcc40b98108229aa54e12449.pdf>

Burry, J., Burry, M., (2010). The New Mathematics of Architecture, Thames and Hudson

Burry, M., (2011). Scripting Culture: Architectural Design and Programming, United Kingdom, Wiley Academy

Campbell-Kelly, M., Aspray, W., Ensmenger, N., Yost, J., R., (2013). Computer: A History Of The Information Machine, United States, Westview Press

Carpo, M., (2011). The Alphabet and The Algorithm, Cambridge, Massachusetts, The MIT Press

Ceccato, C., Hesselgren, L., Pauly, M., Pottmann, H., Wallner, J., (2010). Advances in Architectural Geometry, Wien, Springer

Choi, J., Lee, Y., Lee, E., (2017). Puzzle Based Algorithm Learning for Cultivating Computational Thinking, New York, Wireless Personal Communications, Volume 93-1, 131-145, Erişim <https://link.springer.com/article/10.1007/s11277-016-3679-9>

Choma, J., (2015). Morphing A Guide to Mathematical Transformations for Architects and Designers, London, Laurence King Publishing

Chomsky, N., (1957). Syntactic Structures, Paris, Mouton Publishers, Erişim http://ewan.website/egg-course-1/readings/syntactic_structures.pdf

Chomsky, N., (2001). Dil ve Zihin, (A., Kocaman, Çev.), Ankara, Ayraç Yayınevi

Coates, P., (2010). Programming Architecture, London, Routledge

Corbusier, L., (1948). Modulor, İstanbul, YEM

Corbusier, L., (1955). Modulor 2, İstanbul, YEM

Davis, M., (2016). Algorithms, Equations and Logic, S. B. Cooper & A. Hodges (Edt.), The Once and Future Turing Computing The World, 4-18. United Kingdom, Cambridge University Press

DeLanda, M., (2009). Material Evolvability and Variability, The Architecture of Variation, L. Spuybroek (Edt.), Thames & Hudson, London.

Foucault, M., (2001). Kelimeler ve Şeyler; İnsan Bilimlerinin Bir Arkeolojisi, (M. A. Kılıçbay, Çev.), Ankara, İmge Kitabevi, 36

Futschek, G., (2006). Algorithmic Thinking: The Key for Understanding Computer Science, R. T. Mittermeir (Edt.), Informatics Education – The Bridge Between Using and Understanding Computers, LNCS 4226, 159-168, Berlin, Springer International Publishing, Erişim https://www.researchgate.net/publication/221437678_Algorithmic_Thinking_The_Key_for_Understanding_Computer_Science

Galloway, A. R., Thacker, E., (2006). Language, Life, Code, Collective Intelligence in Design, Architectural Design, 76/5, London, John Wiley

Ganascia, J. G., (2011). Logical Induction, Machine Learning and Human Creativity, T. Bartcherer & R. Coover (Edt.), Switching Codes: Thinking Through Digital Technology In The Humanities and The Arts, 140-158. Chicago and London, The University of Chicago Press

Gibbs – Smith, C., (1978). The Inventions Of Leonardo Da Vinci, Great Britain, Phaidon Press Limited

Gönenç Sorguç, A., Arslan Selçuk, S., (2013). Computational Models In Architecture: Understanding Multi-Dimensionality and Mapping, Nexus, Relationship Between Architecture and Mathematics, Milan, (Vol 15, No 2), 349-362.

Günçe, G., (1971). Jean Piaget ve Temel Kuramsal Fikirleri, Erişim, <http://dergiler.ankara.edu.tr/dergiler/40/488/5718.pdf>

Gürsoy, B., Özkar, M., (2015). Schematizing Basic Design in İlhan Koman's "Embryonic" Approach, Nexus Network Journal, 981-1005.

Henderson, P., B., Cortina, T., J., Hazzan, O., Wing, J., M., (2007). Computational Thinking, SIGCSE (07), 195-196, Erişim <http://dl.acm.org/citation.cfm?id=1227378>

Ifrah, G., (1995). Bir Gölgenin Peşinde Rakamların Evrensel Tarihi (Cilt 1), (K. Dinçer, Çev.), Ankara, Tübitak Yayınları

Jabi, W., (2013). Parametric Design for Architecture, London, Laurence King Publishing

Jiang, T., Li, M., Ravikumar, B., Regan, K. W., (2009). Formal Grammars and Languages, Erişim <http://www.cs.ucr.edu/~jiang/cs215/tao-new.pdf>

Kalay, Y., E., (2004). Architecture's New Media Principles, Theories, and Methods Of Computer Aided Design, Cambridge, The MIT Press, Erişim <https://books.google.com.tr/books?hl=tr&lr=&id=BDboJQJvUq8C&oi=fnd&pg=PP15&dq=vitruvius+man+computational+design&ots=ZrbyiiQyhL&sig=NFmOLf7kW4oMJqGDGMZ8F6N_ags&redir_esc=y#v=onepage&q=vitruvius%20man%20computational%20design&f=false>

Kestelier, X., D. & Peters, B., (2013). Computation Works: The Building Of Algorithmic Thought, Editors. Architectural Design (AD) Journal, Chichester, UK, John Wiley & Sons

Khabazi, Z., (2010). Generative Algorithms: Using Grasshopper

Kim, E. E., Toole, B. A., (1999). Ada and The First Computer, Scientific American, 76-81, Erişim <http://www.cs.virginia.edu/~robins/Ada_and_the_First_Computer.pdf>

Koman, İ., Ribeyrolles, F., (1979). On My Approach To Making Nonfigurative Static and Kinetic Sculpture, Leonardo, (Vol. 12, No. 1), Great Britain, Pergamon Press, 1-4

Kramer, J., (2007). Abstraction: The Key To Computing?, Communications of the ACM, Erişim <https://www.researchgate.net/publication/220427690_Is_abstraction_the_key_to_computing>

Krishnamurti, R., (2006). Explicit Design Space, Pittsburgh, Pennsylvania, Carnegie Mellon University Press

Kuban, D., (1988). Sinan'ın Dünya Mimarisindeki Yeri, Mimarbaşı Koca Sinan, Yaşadığı Çağ ve Eserleri, 581-624, Erişim <http://acikerisim.fsm.edu.tr:8080/xmlui/bitstream/handle/11352/1758/Kuban.pdf?sequence=1>

Leach, N. and Turnbull D., (2004). Digital Tectonics, United Kingdom, Wiley Academy

Legendre, G., L., (2011). The Mathematics Of Sensible Things, Mathematics of Space, Architectural Design, 81/4, London, John Wiley

Leyton, M., (2006). Shape as Memory A Geometric Theory of Architecture, Germany, Birkhauser Publishers for Architecture

Locke, J., (1690). An Essay Concerning Human Understanding, (The Pennsylvania State University, 1999), 589.

Marr, C., D., (1982). Vision: A Computational Investigation Into The Human Represantation And Processing of Visual Information, Massachusetts, MIT Press, 19-25.

Menges, A. and Ahlquist, S., (2011). Computational Design Thinking, United Kingdom, Wiley Academy

Merleau Pounty, M., (1964). The Primacy of Perception, USA, Northwestern University Press

Mitchell, W., J., (1990). The Logic of Architecture: Design, Computation and Cognition, Cambridge, MIT Press

Newton, I., (1687). Principia Mathematica (Vol. 3), (A. Motte, Çev.), New York, Publised Daniel Adee, 384, Erişim <https://docs.lib.noaa.gov/rescue/Rarebook_treasures/QA803A451846.PDF>

Nuriyev, U. G., & Sadıgova, H. G., (2002). Hesaplanabilirlik, Matematik Dünyası, Cilt 11, Sayı 5, 12-16. Erişim <http://www.matematikdunyasi.org/arsiv/PDF_eskisayilar/2002_5_12_16_HESAPLANABILIR.pdf>

Papert, S., (1980). Mindstorms Children, Computers and Powerful Ideas, New York, Basic Books Inc. Publishers

Papert, S., (1996). An Exploration in the Space of Mathematics Educations, International Journal of Computers for Mathematical Learning, (1-1), 95-123.

Peters, B., & Peters, T., (2013). Inside SmartGeometry Expanding the Architectural Possibilities of Computational Design, United Kingdom, John Wiley & Sons Ltd.

Pfeifer, R., & Scheier, C., (2001). Understanding Intelligence, USA, MIT Press, 6.

Picon, A., (2011). Architecture and Mathematics Between Hubris and Restraint, Mathematics of Space, Architectural Design, 81/4, London, John Wiley

Potmann, H., Asperl, A., Hofer, M., Kilian, A., (1990). Architectural Geometry, Singapore, Bentley Institute Press

Randell, B., (Edt.), (1975). The Origins of Digital Computers, New York, Springer – Verlag

Reas, C., Fry, B., (2007). Processing: A Programming Handbook For Visual Designers and Artists, Cambridge, The MIT Press

Reas, C. and McWilliams C., LUST, (2010). *Form + Code in Design, Art and Architecture*, New York, Princeton Architectural Press

Rocker, I., M., (2006). Calculus - Based Form: An Interview With Greg Lynn, Programming Cultures, Architectural Design, 76/4, London, John Wiley

Rocker, I., M., (2006). When Code Matters, Programming Cultures, Architectural Design, 76/4, London, John Wiley

Sağdıç, Z., (2015). Ottoman Architecture: Relationship Between Architectural Design and Mathematics In Sinan's Works, K. William, M. J. Ostwald (Edt.), Architecture and Mathematics from Antiquity To The Future Volume 2, 95-106. Switzerland, Springer International Publishing

Saitta, L., Zucker, J., Zucker, D., (2013). Abstraction in Artificial Intelligence and Complex Systems, New York, Springer

Salvadori, M., (1968). Mathematics in Architecture, USA, Prentice Hall

Salvadori, M., (2015). Can There Be Any Relationships Between Mathematics and Architecture, Architectural Design and Mathematics In Sinan's Works, K. William, M. J. Ostwald (Edt.), Architecture and Mathematics from Antiquity To The Future Volume 2, 25-29. Switzerland, Springer International Publishing

Scheurer, F., Stehling, H., (2011). Lost In Parameter Space, Mathematics of Space, Architectural Design, 81/4, London, John Wiley

Schumacher, P., (2017). Parametricism, Parametricism 2.0: Rethinking Architecture's Agenda For The 21st Century, Architectural Design, 86/2, London, John Wiley

Shannon, C., E., (1948). A Mathematical Theory of Communication, The Bell System Technical Journal, (Vol. 27), 379-423, 623-656.

Shiffman, D., (2012). Nature of Code Simulating Natural Systems with Processing, USA, Free Software Foundation

Sloman, A., (2012). *What is Computational Thinking? Who Needs It? Why? How Can It Be Learnt?*, ALT 2012 Conferance Manchester, Erişim <http://www.cs.bham.ac.uk/research/projects/cogaff/talks/alt2012-sloman.pdf>

Spuybroek, L., (2009). Research & Design, The Architecture of Variation, L. Spuybroek (Edt.), Thames & Hudson, London.

Steadman, J.P., (1983). Architectural Morphology: An Introduction to the Geometry of Building Plans, London, Pion Limited

Stevens, G., (1990). The Reasoning Architect Mathematics and Science in Design, Singapore, McGraw-Hill Book Co

Stiny, G., and Gips, J., (1978). Algorithmic Aesthetics Computer Models for Criticism&Design in the Arts, USA, University of California Press

Stiny, G., (2006). Shape Talking About Seeing Doing, Cambridge, MIT Press

Swade, D. D., (2003). *Calculation and Tabulation in The Nineteenth Century: Airy versus Babbage*, (Doktora Tezi), Erişim <http://ed-thelen.org/bab/Swade_PhD.pdf>

Terzidis, K., (2006). Algorithmic Architecture, USA, Elsevier

Terzidis, K., (2009). Algorithms for Visual Design Using the Processign Language, Indianapolis, Wiley Publishing

Tsigkari, M., Davis, A., Aish, F., (2011). A Sense of Purpose Mathematics and Performance in Environmental Design, Mathematics of Space, Architectural Design, 81/4, London, John Wiley

Turing, A. M., (1936). On Computable Numbers, With an Application To The Entscheidungsproblem,Erişim <https://www.wolframscience.com/prizes/tm23/images/Turing.pdf>

Turing, A. M., (1950). Computing Machinery and Intelligence, Mind, No 236, 433-460., Erişim <https://academic.oup.com/mind/article/LIX/236/433/986238/I-COMPUTING-MACHINERY-AND-INTELLIGENCE>

Yazar, T., (2003). *Mimarlık Eğitiminde Bir Uzman Sistem Modeli Olarak Mimar Sinan Camileri Uzman Sistemi* (Yüksek Lisans Tezi). Erişim <http://www.designcoding.net/decoder/wp-content/uploads/2013/02/2013_02_25-master.pdf>

Williams, M., R., (1990). Early Calculation, W. Aspray (Edt.), Computing Before Computers, 3-58, Iowa, Iowa State University Press

Wing, J. M., (2006). Computational Thinking, Communication of the ACM (49), 33-35, Erişim <http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf>

Wing, J. M., (2008). Computational Thinking and Thinking About Computing, Philosophical Transactions of the Royal Society, (366), 3717-3725.

Vitruvius, (2005). Mimarlık Üzerine On Kitap, (M. H. Moran, Çev.), İstanbul, Şevki Vanlı Mimarlık Vakfı Yayınları

Url-1<https://tr.wikipedia.org/wiki/Ar%C5%9Fimet>, alındığı tarih 17.04 2017

Url-2 <http://www.muhendislikbilgileri.com/?pnum=130&pt=AR%C5%9E%C4%B0MET>, alındığı tarih 22.04 2017

Url-3<http://www.hawking.org.uk/the-origin-of-the-universe.html>, alındığı tarih 28.04.2017

Url-4<https://tr.wikipedia.org/wiki/Jean_Piaget>, alındığı tarih 28.04.2017

Url-5<https://www.youtube.com/watch?v=4oGnvxzOqDw>, alındığı tarih 28.04.2017

Url-6<https://www.youtube.com/watch?v=6-dFTmdX1kU>, alındığı tarih 28.04.2017

Url-7<http://www.bbc.co.uk/education/guides/zp92mp3/revision>, alındığı tarih 21.04.2017

Url-8<http://www.webopedia.com/TERM/C/computational-thinking.html>, alındığı tarih 23.04 2017

Url-9<https://studio.code.org/unplugged/unplug2.pdf>, alındığı tarih 29.04.2017

Url-10<http://cryptomuseum.com/radio/morse/>, alındığı tarih 14.05.2016

Url-11<http://barisguner.com/dilbilim-xxvi-cilt-2-istanbul-2012-aksit-ozturkun-okuma-ugrasi-yapitinin-isiginda-yazinsaldan-muzige-dogru-farkli-bir-bakis-x-esittir-y-c-baglaminda-s-165-173.html/, alındığı tarih 14.05.2016

Url-12<https://www.frmtr.com/kitap-ozetleri/3577866-bu-bir-pipo-degildir-michel-foucault-ozet-analiz.html>, erişim tarihi 07.05.2017

Url-13<http://sanatkaravani.com/gizem-aslinda-hicbir-seydir-rene-magritte/>, erişim tarihi 07.05.2017

Url-14<https://en.wikiquote.org/wiki/Piet_Hein>, erişim tarihi 07.05.2017

Url-15<http://www.piethein.com/page/piet-hein-16/>, erişim tarihi 07.05.2017

Url-16<https://www.arthipo.com/artblog/bilim/leonardo-da-vincinin-bilimsel-calismalari.html>, erişim tarihi 12.05.2017

Url-17<http://www.biography.com/people/groups/academics-mathematicians>, erişim 12.05.2017

Url- 18<https://www.youtube.com/watch?v=aMsaFP3kgqQ>, erişim 12.05.2017

Url-19<http://fihristdergi.com/mahzen/bir-sinir-boyu-yoksunluk/dil-mi-dusunceyi-dogurur-du sunce-mi-dili/>, erişim 21.05.2017

Url-20<https://tr.khanacademy.org/computing/computer-science/informationtheory/moderninfotheory/v/a-mathematical-theory-of-communication>, erişim 21.05.2017

Url-21<https://www.britannica.com/technology/computer/History-of-computing>, erişim 26.05.2017

Url-22<http://www.nms.ac.uk/jacquardloom>, erişim 27.05.2017

Url-23<https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/one.html>, erişim 04.06.2017

Url-24<https://www.youtube.com/watch?v=gJQTFhkhwPA>, erişim 04.06.2017

Url-25<http://bilgisayarkavramlari.sadievrenseker.com/2009/06/25/muntazam-diller-formal-languages/>, erişim 11.06.2017

Url-26
<http://interactivepython.org/courselib/static/pythonds/Introduction/WhatIsProgramming.html>, erişim 30.06.2017

Url-27<https://www.techopedia.com/definition/24815/programming-language>, erişim 30.06.2017

Url-28<https://www.youtube.com/watch?v=D1ELEeIs0j8>, erişim 09.07.2017

Url-29<https://processing.org/tutorials/drawing/>, erişim 14.07.2017

Url-30<https://www.youtube.com/watch?v=OLMxRXwF56w>, erişim 22.07.2017

Url-31<http://mathworld.wolfram.com/MinimalSurface.html>, erişim 22.07.2017

Url-32<http://www.grasshopper3d.com/forum/topics/minimal-surface-3>, erişim 04.03.2017

Url-33<hhtps://www.youtube.com/watch?v=DC6HyZDwFmc>, erişim 27.12.2017

EKLER

EK A: Geometri algoritması (Şekil 4.52)

EK A

Şekil A.1: Geometri algoritması (Şekil 4.52).

ÖZGEÇMİŞ

Ad-Soyad **: Büşra Güler**

Doğum Tarihi ve Yeri **: 02/02/1992, İzmir**

E-posta **:** busra.bguler@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2015, İzmir Ekonomi Üniversitesi, Güzel Sanatlar Fakültesi, Mimarlık Bölümü

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2016 - Melike Altınışık Architects

YAYINLAR, SUNUMLAR VE PATENTLER:

- **Güler, B., Kök, N.** 2016. MSTAS, Mimarlıkta Sayısal Tasarım Ulusal Sempozyumu, Karşılaşmalar: Krizler ve İmkanlar, *moodWall*, Haziran 27-28, 2016 İstanbul, Türkiye.
- **Güler, B., Yasak, H**. 2016. ASCAAD Parametricism Materialism Evolution of Digital Technologies for Development, Uluslararası Konferans, *Experimental Geometry: Redifining Design Using Human Factor,* Kasım 7-8, 2016 London, UK.
- **Güler, B., Yasak, H**. 2016. *Experimental Geometry: Redifining Design Using Human Factor,* ASCAAD Parametricism Materialism Evolution of Digital Technologies for Development, Imperial House Publishers, London, *s.* 275-280.