TÜBİTAK

Mekan Dizimi Tasarım Uygulaması

Program Kodu: 1001

Proje No: 119M082

Proje Yürütücüsü:

Doç. Dr. Pelin DURSUN ÇEBİ

Araştırmacılar:

Y. Mimar Nilüfer KOZİKOĞLU

Doç. Dr. Abdurrahman Tuğrul YAZAR

Danışman:

Mustafa DALCI

Bursiyerler:

Büşra BALABAN

Melike Sena ERDEN

Oğulcan ÜNEŞİ

Ocak 2021

ISTANBUL

ÖNSÖZ

Doç.Dr. Pelin Dursun Çebi (İstanbul Teknik Üniversitesi, Mimarlık Fakültesi) yürütücülüğünde tamamlanan bu çalışma, Y. Mimar Nilüfer Kozikoğlu (Tuşpa Mimarlık), Doç.Dr. Tuğrul Yazar'dan (İstanbul Bilgi Üniversitesi, Mimarlık Fakültesi) oluşan bir araştırmacı ekibiyle gerçekleştirilmiştir. Araş.Gör. Büşra Balaban ve Sena Erden, projede genç araştırmacılar ve bursiyerler olarak görev yapmıştır. Oğulcan Üneşi, araştırma bursiyeri olarak başladığı süreci yazılım geliştirme konusunda hizmet vererek desteklemeye devam etmiştir. Grafik tasarım ve görselleştirme çalışmalarında Emrah Kavlak, Eyes Don't Lie; Unity bazlı oyun yazılımı için Ömer Bilge Ersoy; Yokyer Bilişim Yazılım Araştırma ve Geliştirme Sanayi ve Ticaret Eğitim ve Danışmanlık Ltd Şti; Oyun sahne ve senaryolarının üretilmesi için Emre Kuzuoğlu; Tanıtım çalışmaları için 700 DCC Design Communication Consultancy ile birlikte çalışılmıştır. Araştırma danışmanlığını Mustafa Dalcı yürütmüştür. Tübitak 1001 programı kapsamında gerçekleştirilen bu çalışmanın araştırma ekibine, hizmet veren paydaşlarına, atölyelere ve bilimsel toplantılara katılarak değerlendirmeleriyle bize yol gösteren sevgili öğrencilerimize, değerli öğretim üyelerimize özverili çalışmaları ve katkıları için sonsuz teşekkürlerimizi sunarız.

Ocak 2021

İÇİNDEKİLER

ÖNSÖZ i

İÇİNDEKİLER ii

TABLO LİSTESİ v

ŞEKİL LİSTESİ v

ÖZET viii

ABSTRACT ix

1. GİRİŞ 1

2. LİTERATÜR ÖZETİ 3

3. GEREÇ VE YÖNTEM 13

4. BULGULAR 15

4.1 SpaceChase for Grasshopper 15

4.1.1 Hedefler / Adımlar 15

4.1.2 Geliştirme Süreci 17

4.1.3 Temel Kullanım Biçimi 34

4.1.4 Sınırlar ve Olanaklar 42

4.2 SpaceChase Oyunu 44

4.2.1 Hedefler / Adımlar 45

4.2.2 Geliştirme Süreci 46

4.2.3 Temel Kullanım Biçimi 61

4.2.4 Sınırlar ve Olanaklar 66

4.3 Yazılım Testleri ve Geri Bildirimler 67

4.3.1 Atölyeler ve Yazılım Testleri 67

4.3.2 Bilimsel Toplantılar 74

4.3.3 Web Sitesi Üzerine Değerlendirmeler 78

5. TARTIŞMA / SONUÇ 81

5.1 Yazılımlara İlişkin Karşılaştırmalı Okumalar 81
5.1.1 Modeller 82
5.1.2 Ortamlar ve Kullanıcılar 83
5.1.3 İşlevler 83
5.1.4 İşleyiş Mekanizmaları ve Süreçler 84
5.2 Öneri ve Değerlendirmeler 86

KAYNAKLAR 90

EKLER 94
EK 1. SpaceChase for Grasshopper WIP Prototip #2 Veri Akış Diyagramı 95
EK 2. SpaceChase for Grasshopper WIP Prototip #3 Veri Akış Diyagramı 98
EK 3. SpaceChase for Grasshopper WIP Prototip #4 Veri Akış Diyagramı 102
EK 4. SpaceChase for Grasshopper WIP Prototip #4 Ekranları 106
EK 5. SpaceChase for Grasshopper Beta (0.1.0) Geliştirilen Bileşenler 110
EK 6. SpaceChase for Grasshopper Beta (0.1.0) Örnek Veri akış Diyagramı 112
EK 7. SpaceChase for Grasshopper Beta (0.2.5) Geliştirilen Bileşenler 115
EK 8. SpaceChase for Grasshopper Beta (0.2.5) Örnek Veri akış Diyagramları 118
EK 9. SpaceChase for Grasshopper Beta (0.3.3) Geliştirilen Bileşenler 120
EK 10. SpaceChase for Grasshopper Beta (0.3.3) Örnek Veri Akış Diyagramları 124
EK 11. SpaceChase for Grasshopper Değişim Günlüğü (Changelog) 127
EK 12. SpaceChase for Grasshopper Temel Kullanım Kılavuzu 130
EK 13. SpaceChase Oyunu Arayüz Etkileşimini Gösteren Örnek Ekran Görüntüleri 130
EK 14. Oyun Elemanları İlişkilerinin Anlatım Diyagramı 156
EK 15. SpaceChase Oyunu Kullanım Kılavuzu 158
EK 16. Atölye I Programı 172
EK 17. Atölye II Programı 173
EK 18. Atölye III Programı 174
EK 19. Atölye IV-V Programı 176
EK 20. Terimler Dizini 177

TABLO LİSTESİ

Tablo 1. SpaceChase for Grasshopper prototiplerinde kullanılan örnek .csv tablosu. ……..19
Tablo 2. “SpaceChase for Grasshopper WIP” prototiplerinin özelliklerinin karşılaştırılması. ……...24
Tablo 3. Space Chase oyunu örnek sahneler için parametre ayarları. ……60
Tablo 4. Space Chase oyunu örnek sahne çalışması yönetici ekranı. ……71

ŞEKİL LİSTESİ

Şekil 1. Mevcut Mekan Dizimi yazılımlarının incelenmesi. ……..8
Şekil 2. SpaceChase for Grasshopper Work-in-progress (WIP) Prototip #1 (Ekim 2019). Veri akış diyagramı. ……..17
Şekil 3. SpaceChase for Grasshopper WIP Prototip #1. Ekran görüntüsü. Kullanıcı, düğüm noktalarını fare yardımıyla sürükleyip itme ve çekme hareketlerini tetikleyebilmektedir. ……18
Şekil 4. "SpaceChase for Grasshopper WIP" Prototip #2 (Aralık 2019). Veri akış diyagramı. Bu diyagram daha detaylı olarak EK 1'de verilmiştir. ……18
Şekil 5. "SpaceChase for Grasshopper WIP" Prototip #2 (Aralık 2019). Ekran görüntüsü. ….20
Şekil 6. "SpaceChase for Grasshopper WIP" Prototip #3 (Aralık 2019). Veri akış diyagramı. Bu diyagram daha detaylı olarak EK 2'de verilmiştir. ……21
Şekil 7. "SpaceChase for Grasshopper WIP" Prototip #3. Ekran görüntüsü. ……21
Şekil 8. "SpaceChase for Grasshopper WIP" (Haziran 2020). Prototip #4; veri akış diyagramı. Bu diyagram daha detaylı olarak EK 3'te verilmiştir. ……22
Şekil 9. "SpaceChase for Grasshopper WIP" Prototip #4. Genel ekran görüntüsü. Bu prototipin diğer ekranları EK 4'te sunulmuştur. ……24
Şekil 10. "SpaceChase for Grasshopper Beta" (0.1.0) (Ağustos 2020). Eklenti sekmesi ve bileşen grupları. Bu bileşenlerin ayrıntılı açıklamaları EK 5'te sunulmuştur. ……26
Şekil 11. "SpaceChase for Grasshopper Beta" (0.1.0); örnek veri akış diyagramı. Bu kod Ek 6'da ayrıntılı olarak verilmiştir. ……28
Şekil 12. "SpaceChase for Grasshopper Beta" (0.1.0). Örnek ekran görüntüsü. ……..29
Şekil 13. "SpaceChase for Grasshopper Beta" (0.2.5) (Eylül 2020). Eklenti sekmesi ve bileşen grupları. Bu bileşenlerin ayrıntılı açıklamaları EK 7'de sunulmuştur. ……..30
Şekil 14. "SpaceChase for Grasshopper Beta" (0,2,5); örnek veri akış diyagramı ve ekran görüntüsü. ……31
Şekil 15. "SpaceChase for Grasshopper Beta" (0.2.5). Ekran görüntüsü. Eklentinin görselleştirme kapasitesinin test edildiği farklı denemeler Ek 8'de sunumuştur. ……31
Şekil 16. "SpaceChase for Grasshopper Beta" (0.3.3) (Kasım 2020). Eklenti sekmesi ve bileşen grupları. Bu bileşenlerin ayrıntılı açıklamaları EK 9'da sunulmuştur. ……32
Şekil 17. "SpaceChase for Grasshopper Beta" (0.3.3); örnek veri akış diyagramı. Bu kod Ek 10'da daha ayrıntılı olarak verilmiştir. ……...32
Şekil 18. "SpaceChase for Grasshopper Beta" *Advanced Graph* modelinin sunduğu "*Contextual Node Manager*" ve "*Link Manager*" arayüzleri. ……..33
Şekil 19. SpaceChase sekmesinde bulunan bileşenler 7 grup içerisinde organize edilmiştir. ……..35

Şekil 20. *Construct Graph* bileşeni ile düğüm (ND) ve bağlantı (E) verilerinden SpaceChase düğüm (N) ve ağ (GD) verilerinin türetilmesi. ……..……35
Şekil 21. Düğüm noktalarının görselleştirilmesi ve entegrasyon değerlerinin okunması. …..36
Şekil 22. *Justified Tree Graph* oluşturulması. ……36
Şekil 23. Bağlamsal düğüm noktalarının, negatif bağlantıların (kırmızı) ve bağlamsal bağlantıların (mavi) tanımlanması ve Rhinoceros'ta farklı katmanlarda çizilmesi. ……..…..37
Şekil 24. *Construct Advanced Dynamic Graph* bileşeninin temel bağlantıları. ……..38
Şekil 25. *Construct Contextual Node* bileşeni ve *Contextual Node Manager* penceresi. ….....38
Şekil 26. *Construct Advanced Dynamic Graph* bileşeninin bağlantıları ve *Link Manager* penceresi. ……39
Şekil 27. Kangaroo *Bouncy Solver* bileşenine girdilerin aktarılması ve hareketli ağ için ayarlanması. ……39
Şekil 28. *Contextual Node Manager*'in hareketli ağ üzerinde gerçek zamanlı olarak kullanılması. ……..……..40
Şekil 29. Link Manager arayüzü kullanılarak düğüm noktaları arasındaki ilişkilerin daha hassas biçimde kontrol edilmesi. ……..……40
Şekil 30. Hareketli ağ yapısına elle müdahale etmek mümkündür. Bunun için fare ile düğüm noktaları tutulup sürüklenmelidir. ……..41
Şekil 31. SpaceChase örnek dinamik ağ kurulumu. ……42
Şekil 32. Oyun tasarım alanının temel alt başlıkları. ……..46
Şekil 33. Yapboz çözümü ve ağ modellemesinin etkin olduğu var olan oyun kaynaklarının incelenmesi. ……48
Şekil 34. İşlevleri ile tanımlanmış mekansal birim örnekleri. ……..……..49
Şekil 35. Farklı oyun mantıklarından oyuncu başarı motivasyonu karakteristikleri. ……50
Şekil 36. *Miro* dijital arayüzünde proje ekibin gerçekleştirdiği ortak eskiz örneği. …...……..51
Şekil 37. Oyun tasarım sürecine dair üretimlerden örnekler. ……51
Şekil 38. Dijital eskiz örnekleri. ……52
Şekil 39. Görsel esin kaynakları örnekleri. ……..52
Şekil 40. Oyun arayüzünü meydana getiren iki ana bardan biri (üst blok). ……53
Şekil 41. Oyun arayüzünü meydana getiren iki ana bardan biri (alt blok). ……54
Şekil 42. Oyunu meydana getiren adımlar şeması. ……..55
Şekil 43. Düğüm çeşitleri için görsel tasarım örnekleri. ……56
Şekil 44. Bağlamsal düğümler (actuator) için görsel tasarım örnekleri. ……..57
Şekil 45. Newton'ın evrensel kütleçekimi yasasının formülasyonu. ……60
Şekil 46. Prototip sistem akış diyagramı. ……60
Şekil 47. Ön prototip uygulamanın örnek sahne çalışması anından ekran görüntüsü. ……61

Şekil 48. SpaceChase ekran akış diyagramı. ……63
Şekil 49. Oyunun bölüm yapısı akış diyagramı. ……64
Şekil 50. Oyuncunun tercihleri ile süren örnek sahne ve bölüm akış diyagramı. ……..66
Şekil 51. Atölyelere ilişkin veri analiz şeması. ……68
Şekil 52. İkide1 Oyunu: oyun anından görüntüler. ……69
Şekil 53. Atölye I'de incelenen üç programa ilişkin karşılaştırmalı katılımcı değerlendirmeleri. ……69
Şekil 54. SpaceChase for Grasshopper ile katılımcılar tarafından üretilen sahne örnekleri.….. ……70
Şekil 55. Atölyelerde öne çıkan katılımcı değerlendirmelerinin dökümü. ……..73
Şekil 56. Üniversitelere göre katılımcı profili. ……..74
Şekil 57. Bilimsel toplantı serisi içerik şeması. ……..76
Şekil 58. Bilimsel toplantılar kapsamında öne çıkan değerlendirmeler. ……78
Şekil 59. SpaceChase Game Instagram hesabı, Aralık ayı kullanıcı analizi. ……80
Şekil 60. Yazılım setlerinin bileşenleri. ……81
Şekil 61. Yazılım/ların hedefleri. ……...83

ÖZET

Mekan Dizimi, mimarlıkta ağ düşüncesine odaklanan, grafik-teorik araçlarla mekansal biçimlenmelerin ürettikleri potansiyelleri deşifre etmeye çalışan, mimarlık ve yapılı çevreye ilişkin bilimsel, veri odaklı, teorik bir kavrayış sunmayı hedefleyen bir teknikler bütünüdür. Mekan Dizimi araştırmalarının mimari tasarımda kullanılabilirliği ilgi çekici olmakla birlikte, bu tür bilimsel araçların tasarım sürecinde üretken bir biçimde kullanımına yönelik çalışmalar sınırlıdır. Araştırma projesi, bu boşluğu doldurmayı hedeflemekte, Mekan Dizimi'nin gerçek zamanlı bilgi üreten, dinamik, yaratıcı bir araç olarak mimari tasarım sürecinde kullanılabilirliğini sorgulamaktadır. Bu bağlamda projenin ana ilgisi, Mekan Dizimi kuram ve uygulamalarından yola çıkan, topolojik-geometrik mekansal kurgular türetebilen, bilimsel temelli ve oyunsallaştırılabilir bir bilgisayar yazılımının geliştirilmesi üzerinedir.

Bu proje yöntem olarak öncelikle kuramsal altyapıyı zenginleştirmek üzere mekansal ağları deşifre etmeyi hedefleyen Mekan Dizimi odaklı mevcut bilgisayar yazılımlarının olanak ve sınırlamalarını irdeler. Ardından yeni yazılım/lar için kodlama çalışmalarına, ilgili kriter/algoritma ve simülasyonların gerçekleştirilmesine odaklanır. Geliştirilen "oyunsallaştırılmış yazılım/lar", mimarlık öğrencileriyle ve araştırma grubu içinden ve dışından uzman kişilerle gerçekleştirilen atölye çalışmalarıyla test edilmiş, potansiyelleri ve sınırları değerlendirilmiştir.

Araştırma kapsamında iki yazılım geliştirilmiştir: *SpaceChase for Grasshopper* ve *SpaceChase.* SpaceChase for Grasshopper, Mekan Dizimi ile ilişkili metriklerin dinamik ağ yapısı üzerindeki etkisini araştırabilmek üzere geliştirilen, mimar ve mimarlık öğrencisi kullanıcılara yönelik olarak tasarlanan, *Rhinoceros / Grasshopper* yazılımı içerisinde çalışan bir eklentidir; "oyunsal yazılıma" bir ön hazırlık olarak düşünülebilir. Bu eklentiye paralel olarak tasarlanan, mimarlar dışında daha geniş bir kullanıcı kitlesine de ulaşmayı hedefleyen *SpaceChase* ise mekansal birimleri ve bağlama ait verileri düğümler aracılığıyla temsil eden, aralarındaki ilişkileri kurup bozarak olası bir araya gelmeleri, konfigürasyonları araştıran ve arka planda tüm olasılıkları Mekan Dizimi metrikleri ile değerlendirilebilir ve ölçülebilir hale getiren, bilim odaklı oyunsal bir yazılımdır.

Akademiden ve uygulamadan çeşitli uzmanların ortaklığında gerçekleştirilen ve mekansal organizasyonların ölçülmesine, bir tasarım araştırması olarak ele alınmasına odaklanan deneysel çalışmanın mimarlık eğitimi ve pratiği için özgün bir değer ürettiği düşünülmektedir.

Anahtar Kelimeler: Mekan Dizimi, Kanıt Temelli Tasarım, Mimarlıkta Ağ Düşüncesi, Mimari Tasarım

ABSTRACT

Space Syntax as a research realm is a set of techniques aimed at presenting scientific, data-oriented, theoretical understanding of architecture and built-environment in this sense, focuses on architectural thinking, deciphering the potentials of spatial formations through graphic-theoretical digital tools. Space Syntax intends to understand and analyze the existing spatial constructions. However, limited experimental studies are investigating the topological-geometrical probabilities of spatial configurations in recent years. This research intends to enrich the mentioned efforts and possibilities in this limited area of interest and investigates the potential use of the conceptual framework in Space Syntax as a creative tool that generates real-time information in the architectural design process. The main purpose of the project is to produce software investigating the use of graph-based thinking in the architectural design process that generates topological and geometric layout possibilities.

This work first examines the possibilities and limitations of existing Space Syntax-oriented computer software that intends to decrypt and measure spatial networks/configurations. Then it focuses on coding studies for two new software, realization of relevant criteria/algorithms and simulations. Developed "game-like software" was tested in several workshops and their possibilities and limitations were evaluated with architectural students, young architects and experts.

Two software have been developed within the scope of the research; *SpaceChase for Grasshopper* and *SpaceChase.* SpaceChase for Grasshopper as a plug-in that works within Rhinoceros / Grasshopper software, designed for architecture students and architects and developed to investigate the effect of Space Syntax metrics on dynamic graph. Designed in parallel with this plug-in and aimed to reach a wider user group, SpaceChase is a science-oriented game-like software that represents spatial units and context data through the nodes, investigates possible combinations, configurations by establishing and disrupting the relationships between these nodes and makes all possibilities evaluable and measurable with Space Syntax metrics in the background.

It has been considered that this experimental work, which is carried out in partnership with various experts from academy and practice, and which puts forward to measure spatial organizations and to use them as a design research creates a unique value for both architectural education and architectural practice.

Keywords: Space Syntax, Evidence-Based Design, Network Thinking, Architectural Design

1. GİRİŞ

Mekan Dizimi insanların mekanı nasıl kurguladıkları ve nasıl kullandıklarına ilişkin teorik bir kavrayış geliştirerek mimarlıkta konfigürasyonel bir teori inşa etmeye yönelik bir girişimdir. Burada ana ilgi mekana gömülü olan sosyal bilginin (kurallar ve anlamlar), matematik ve grafik araçlar kullanılarak ölçülebilir bir forma dönüştürülmesi, görünür ve üzerinde konuşulabilir kılınması üzerinedir. Bilimsel, analitik, veri odaklı yaklaşım sergileyen Mekan Dizimi'nin mimari tasarım süreci içinde kullanılabilirliği ilgi çekici bir konu olmakla birlikte bu alanda yapılan çalışmalar sınırlıdır. Bu araştırma, mimari tasarım sürecinin denemeler üzerine kurulu bir araştırma, keşif süreci olduğuna vurgu yaparak Mekan Dizimi özelinde bilimsel verinin bu süreci bulgularıyla besleyen, mimarlar için "onunla birlikte düşündüğümüz" bir araç olarak nasıl dahil olabileceğini sorgulamaktadır.

Mimari tasarım öznel bir süreç olmakla birlikte, kimi tasarım araçları ve metodları, tasarımcıya tasarlananı değerlendirmek ve öğrendikleriyle yeniden üretmek için nesnel kriterler sunar. Ağ düşüncesi içeren mekan kurma ve ölçme araçları, mimari tasarım kararlarını görselleştirmeye, mimari programa yönelik potansiyelleri ortaya çıkarmaya ve senaryolar geliştirmeye olanak verir. Bu proje, mimari tasarımda ağ düşüncesinin kullanılmasına odaklanarak, bilimsel veri ve süreçlerin tasarım sürecine nasıl taşınabileceğini, bu tür bir düşünme biçiminin tasarım sürecinde yaratıcı ve bilgilendirici bir araç olarak nasıl işlevselleşebileceğini araştırmaktadır. Ana ilgi, bilimsel veriyi tasarım sürecine taşıyarak "kanıt temelli tasarım" sürecinin özünü biçimleyen Mekan Dizimi'nin mimari tasarım sürecinde gerçek zamanlı bilgi üreten, dinamik, yaratıcı bir araç olarak kullanılabilirliğinin sorgulanması üzerinedir. Bu hedefle *Mekan Dizimi Tasarım Uygulaması* adlı araştırma projesi, topolojik-geometrik mekansal kurgular türetebilen, bilimsel temelli ve *oyunsallaştırılabilir* bir bilgisayar yazılımının geliştirilmesine odaklanır.

Proje kapsamında çalışmanın kuramsal arka planını kurgulamak amacıyla mekansal ağları deşifre etmeye niyet eden Mekan Dizimi odaklı mevcut bilgisayar yazılımlarının olanak ve sınırlamaları irdelenmiştir. Analitik özellikler ötesinde dinamik ve etkileşimli bir karaktere sahip, tasarımcının çok yönlü düşünmesine ve üretmesine destek veren yeni nesil bir yazılımın varlığı sorunsallaştırılmıştır. Bu değerlendirmeler sonucunda Mekan Dizimi'nin ortaya koyduğu teorik yaklaşım ve araç setinden yararlanarak mimara tasarımlarının mekan kaliteleri hakkında gerçek zamanlı, bilimsel, objektif veriler sunan bir bilgisayar yazılımının geliştirilmesi üzerine odaklanılmıştır. Bu yazılımın "*oyunsallaştırılabilir*" bir akıl içermesi ve tasarımcının/mimarın kolayca kavrayabileceği, kullanıcı dostu bir kurgu ortaya koyması önemsenmiştir. Bu yazılımı geliştirme sürecinde, taslak senaryolar üzerinde çalışarak dinamik ağ yapısının hareket

biçimlerini test etmek ve Mekan Dizimi ile ilişkili metriklerin dinamik ağ üzerindeki etkisi ile ilgili çalışmaları yürütmek amacıyla bir ön çalışmanın yararlı olacağına karar verilmiştir. Mimarlara ve mimarlık öğrencilerine yönelik olarak tasarlanan, *Rhinoceros/Grasshopper* yazılımı içerisinde çalışan ve *SpaceChase for Grasshopper* adı verilen bu eklentinin araştırmaya konu olan "*oyunsal yazılıma"* bir ön hazırlık olması, bulgularıyla oyunsal yazılımı beslemesi planlanmıştır. Bu eklentiye paralel olarak tasarlanan, mimarlar dışında daha geniş bir kullanıcı kitlesine de ulaşma hedefiyle kapsamını güncelleyen *SpaceChase* adındaki oyunsal yazılım ise mekansal birimleri ve içinde bulundukları bağlama ait verileri düğümler aracılığıyla temsil ederek ve aralarındaki ilişkileri kurup bozarak, olası bir araya gelmeleri, konfigürasyonları araştırmayı amaçlamaktadır. Ön planda tutulan oyunsal arayüze, arka plandaki oyun mekanlarına ilişkin sentaktik veriler de eşlik etmektedir.

Proje kapsamında SpaceChase for Grasshopper ve SpaceChase Oyunu'nun yazılım testleri araştırma grubu içindeki uzmanların katılımıyla gerçekleştirilmiş, bu denemeler sürece yayılan atölyelerden de beslenmiştir. Gerçekleştirilen bilimsel toplantılar geleceğe yönelik yeni hedefler/açılımlar için yol gösterici olmuştur. Testlerle sürekli gelişim gösteren eklenti ve oyunun güncel sürümlerine ilişkin tartışmalar ve değerlendirmeler ilgili bölümlerde detaylı olarak ele alınmaktadır.

Mimari tasarım sürecinde mekansal ilişkiler üzerine kurulu, eskiz benzeri bu oyunsal aracın, yaptıklarından öğrenen mimarlar için yeni bakış açıları üreteceği, mimarlık dışındaki kullanıcı kitlesi için ise mekan üzerine düşünmede önemli bir farkındalık geliştireceği düşünülmektedir. Aynı zamanda sözü edilen oyunsal araç, bilimsel verinin tasarım sürecine taşınması, gerçek zamanlı, dinamik, etkileşimli bir arayüz kurgulaması anlamında mimari tasarım eğitimi ve pratiği için özgün bir değer üretmektedir.

2. LİTERATÜR ÖZETİ

Mimari tasarım, yapılarak öğrenilen, karmaşık, bilişsel, entelektüel bir süreçtir (Cross, 2001, 2007). Yapma pratiğine odaklı bir tür keşif, araştırma, inceleme, öğrenme sürecini içerir (Dursun, 2007). Optimal bir çözüm bulma yerine (Simon, 1996), birçok değişkenin birbiriyle ilişkilendirilip yorumlandığı, buna bağlı olarak düşüncelerin sürekli yapılandırıldığı ve mekansallaştırıldığı bir süreç içerir. Bu anlamda, “yaparak öğrenilen” (Schön, 1987), dinamik, doğrusal olmayan, karmaşık ilişkileri barındıran performatif bir süreç olarak düşünülebilir. Bu süreçte farklı tasarım araçlarıyla düşünen mimar (Dursun ve Kozikoğlu, 2017), kendine özgü bir yaklaşım, tasarımcıya özgü bir bilme, yapma ve düşünme hali geliştirir (Cross, 2001, 2007). Başka bir deyişle mimar, farklı tasarım araçlarını ve bilgi kaynaklarını ustaca kullanabilen ve birbirleriyle ilişkili olarak farklı bilgileri toplayıp yorumlayarak yeni tasarım düşünceleri üretebilen entelektüel bir kişidir (Dursun, 2007).

Mimarlıkta morfolojik çalışmaların, bir başka deyişle biçim bilimsel yaklaşımların tasarım sürecine nasıl entegre edilebileceği önemli bir araştırma konusudur. Bu tür yaklaşımlar ile mekanı tasarlamak ve ölçmek, mimari kararların görselleştirilmesine, tasarlanan mekanların potansiyellerinin araştırılmasına, mimari program ve tasarım senaryolarının yeniden yapılandırılmasına olanak vermektedir. Burada ana ilgi bilimsel verilerin ve süreçlerin tasarım sürecinde tasarım düşüncesini zenginleştiren yaratıcı, bilgilendirici, üretken bir araç olarak nasıl kullanılabileceği üzerinedir.

Mimari tasarımda bilimsel verinin/araçların kullanımı, mimarın sezgisini baskılamayı değil, aksine onu sorgulanabilir, üzerinde konuşulabilir hale getirmeyi amaç edinir. Mimaride ağ düşüncesine odaklanan, bilimsel bir yaklaşım olan Mekan Dizimi (Hillier, 1996), mekanların bir araya getirilme ve kullanıcılar arasında belirli bir etkileşim modeli ortaya koyma şeklinin birtakım sosyal-kültürel anlamlar gösterdiğini ifade etmektedir (Hillier, 1996). Mekan Dizimi, mekanın nasıl kurgulandığı ve nasıl kullandığına ilişkin bilimsel veriler üretirken, bu veriler ile tasarımcının tasarladıkları üzerinde düşünübilmesine de olanak verir. "Mekan Dizimi araştırması nedene dayalıdır ve çoğundan daha titizdir, ancak yarattıkları aracılığıyla mimari sezginin incelenmesine etkili bir şekilde yol açmıştır. Uygulamada tasarım, sezgi ve mantığı karıştırarak ilerler. Mekan Dizimi, sözle ifade edilemeyen sezgiyi daha rasyonel ve dolayısıyla sözle ifade edilebilir hale getirir.” (Hillier ve Hanson, 1997).

Yaygın literatür incelendiğinde Mekan Dizimi’nin tasarım sürecine katkısı şu şekilde özetlenebilir (Dursun, 2007):

A. “Mimar ve tasarlanmış mekan arasındaki diyalogda Mekan Dizimi, mekan üzerine düşünmek ve konuşmak için bir dil sunar. Bu mimarların alışık olmadığı bir dildir; daha matematiksel ve bilimseldir. Ancak mekanın sözle ifade edilemeyen özelliklerini sözle ifade edilebilir hale getirip onu tartışmaya açar.

B. Bilimsel veriyi tasarım sürecine taşır. Araştırma ve tasarım arasındaki bağı kurar ve bu şekilde “kanıt temelli tasarım”ın özünü biçimler.

C. Eğer tasarım, yaparak ve test ederek öğrenilen bir aktivite ise Mekan Dizimi bu sürece mimarın düşüncelerini araştırmak, tasarımlarının olası etkilerini anlamak, yaptıklarının nasıl çalışacağını görmek için bir dizi araç sunar.”

Bu araçlar, mekanın temsili ve analizi için geliştirilen teknikler, ölçüm için kullanılan sentaktik metrikler ve bilgisayar yazılımlarıdır. Mekan Dizimi araştırmaları, mekansal analizler için geliştirilen birçok bilgisayar programından yararlanmaktadır. Bu yazılımlar ile mekansal analizler, sayısal ve grafik anlamda görünür kılınmaktadır. Bu yazılımların 1980’li yıllardan günümüze zenginleştiği, teknolojik gelişmelere bağlı olarak farklı ortam ve kalitelerde işlevselleştiği görülmektedir. Bugün söz konusu sürecin çeşitli araştırmalar ekseninde devam ettiği söylenebilir. Bu araştırma projesinin de ana ilgisi, tasarım sürecinin işleyişine vurgu yaparak bu programların sorunsallaştırılması, mevcut araç setinin bu sürece eşlik edebilecek kalitelerle zenginleştirilmesi üzerinedir. Proje kapsamında geliştirilmesi amaçlanan “Oyunsallaştırılmış Mekan Dizimi Yazılımı”nın üretilme sürecinde ilk adım mevcut Mekan Dizimi bilgisayar yazılımlarının irdelenmesi, bu programların olanak ve sınırlamalarının, işleyiş mekanizmalarının anlaşılması olarak belirlenmiştir. Araştırma kapsamında ulaşılabilen programların analizi üzerinden elde edilen veriler ile projeye konu olan yazılımın geliştirilmesine katkı sağlanması hedeflenmiştir.

Bugüne kadar geliştirilen ve araştırma kapsamında ulaşılabilen Mekan Dizimi yazılımları incelendiğinde; Nick Dalton tarafından 1988 ve 2004 yıllarında geliştirilen *Axman* ve *WebMapatHome*, Alasdair Turner tarafından ilk kez 2001’de ortaya konan *Depthmap* (Turner, 2007), yine UCL’de (University College of London) geliştirilen N*etBox, Pesh, Omnivista, NewWave, Space Box and Orange Box*, Lucas Figueiredo tarafından geliştirilen *Mindwalk*, Daniel Koch tarafından 2004 yılında geliştirilen *Jass,* Shinichi Iida tarafından 2005 yılında geliştirilen *Segmen*, Henrik Markhede ve Pablo Miranda Carranza tarafından geliştirilen *SPOT* (Markhede ve Carranza, 2007) gibi birçok bağımsız olarak kullanılabilen bilgisayar yazılımının olduğu görülmektedir.

Sözü edilen bu programlar içinde Daniel Koch tarafından 2004 yılında geliştirilen Jass, dışbükey mekan analizi için kullanılan, basit arayüze sahip bir yazılımdır. Bu yazılım kullanıcının, üzerinde bir düğüm ağ grafiği çizmek için programın içine görseller aktarmasına izin vermektedir ve bu grafik üzerinden analiz yapmaktadır. Mekansal birimleri ifade eden düğümler ve aralarındaki ilişkiler belirtilerek Mekan Dizimi analizlerinin yapılması sağlanmaktadır. Tanımlayıcı bir ürün çıktısı oluşturan bu program mekansal analiz sonucunda, geçiş grafikleri oluşturmakta ve Mekan Dizimi metriklerini hesaplamaktadır (Koch, 2004).

Alasdair Turner tarafından ilk kez 2001'de ortaya konan Depthmap ise (Turner, 2007) mekansal konfigürasyonların görünürlük grafiklerinin oluşturulması ve analiz edilmesi için kurgulanmıştır. Program zaman içinde geliştirilmeye devam edilmiş ve UCL'deki Space Syntax Laboratory (Mekan Dizimi Laboratuvarı) tarafından son sürümü olan *DepthmapX* ortaya konmuştur. DepthmapX, yapılı çevredeki sosyal süreçleri anlamak için tasarlanmış bir dizi Mekan Dizimi analizi gerçekleştirebilen açık kaynaklı bir yazılım platformudur. Binalardan kentlere veya eyaletlere kadar çeşitli ölçeklerde çalışmaktadır. Her ölçekte, yazılımın amacı, açık alan öğelerinin bir haritasını oluşturmak, bunları bir ilişki yoluyla (örneğin; görülebilirlik [intervisibility] veya örtüşme [overlap] ile) birbirine bağlamak ve ardından ortaya çıkan ağın analizini yapmaktır (URL-1). Bu programın konfigürasyonel yapıların analizini ortaya koymakla birlikte bulgularıyla tasarımcıya tasarladıkları üzerinden geri bildirim veren bir kaliteye sahip olduğu söylenebilir.

Öte yandan mevcut programların bir bölümünün ise bir program uzantısı olarak çalıştığı görülmektedir. Örneğin Bing Jiang tarafından 1998'de geliştirilen *Axwoman*, Nick Dalton tarafından 2001'de geliştirilen *Ovation*, Michael Batty tarafından 2003 yılında geliştirilen *Ajax*, Alexander Ståhle ve ekibince geliştirilen *Place Syntax Tool* ve Mekan Dizimi Limited Şirketi tarafından geliştirilen *Confeego* ve *iVALUL* (Gil ve diğ. 2015) gibi programlar yalnızca *Coğrafik Bilgi Sistemleri (GIS)* uzantısı olarak çalışmaktayken, John Peponis'in Georgia Teknoloji Enstitüsü'ndeki araştırma grubunca geliştirilen *Spatialist*, Michigan Üniversitesi tarafından geliştirilen *Sytanx2D* gibi bir kısım programlar ise yalnızca *Bilgisayar Destekli Tasarım (CAD)* uzantısı olarak çalışmaktadır. Alain Chiaradia ve arkadaşlarınca geliştirilen *sDNA* (Cooper, Chiaradia, 2015) ve Sevtsuk ve Mekonnen tarafından geliştirilen *UNA Toolbox* (Sevtsuk ve Mekonnen, 2012) ise hem CAD hem GIS ortamlarına uyumlu olarak çalışabilmektedir.

Ayrıca yaklaşık son on yıldır Grasshopper gibi Mekan Dizimi analizlerinin üç boyutlu tasarıma imkan veren CAD ortamlarına entegre edilmesine ve üretken tasarım sürecinin bir parçası olmasına izin veren tasarım araçları için çeşitli eklentiler ve araç setleri de geliştirilmektedir. Örneğin; bunlar arasında eksensel çizgiler ve segment haritalar ile çalışan *Spiderweb*

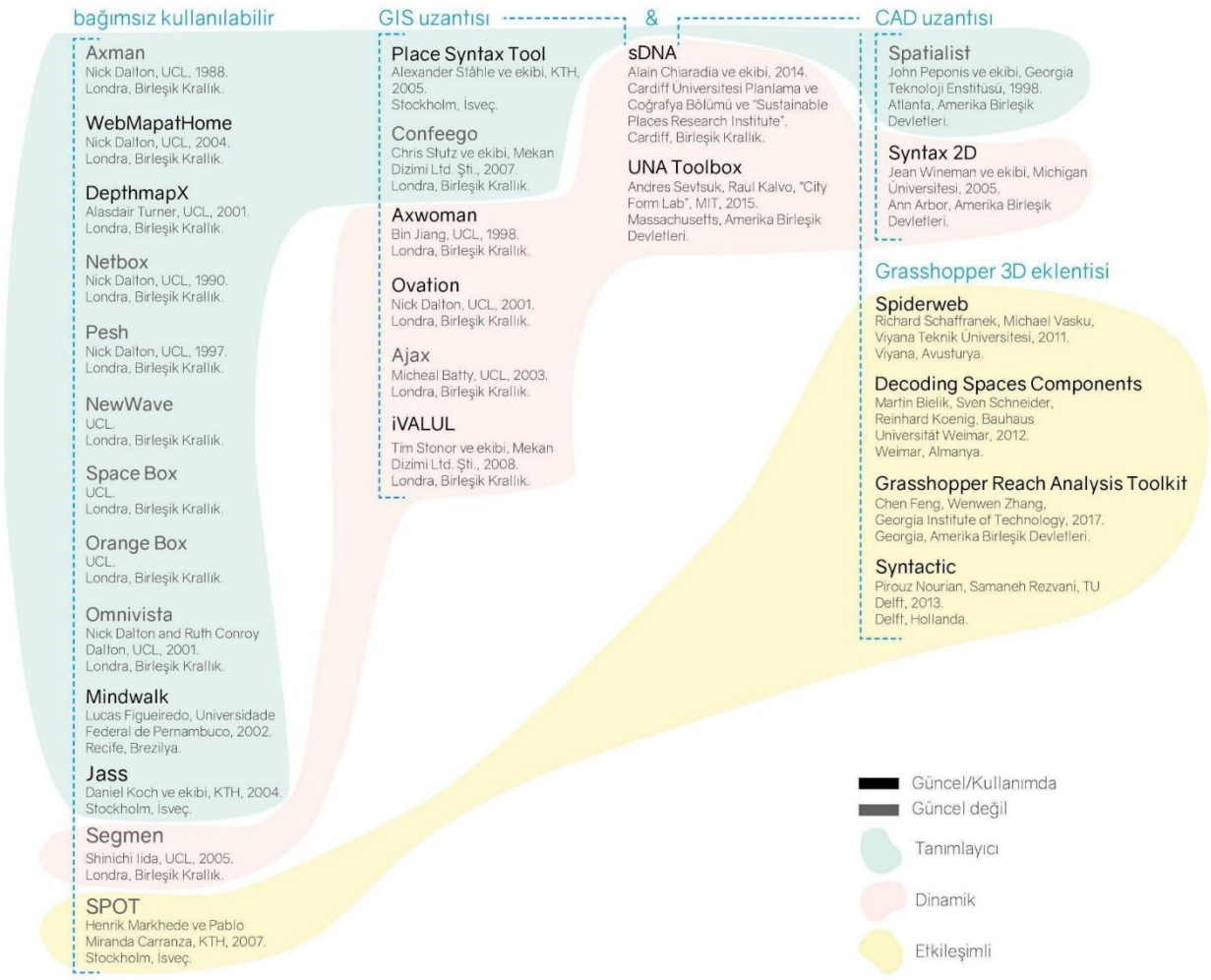
(Schaffranek ve Vasku, 2013) ve *Decoding Spaces Components* (Bielik, et al., 2012) gibi eklentiler, ağırlıklı olarak kentsel tasarım alanında kullanılmakta ve Mekan Dizimi analizlerini, mimari formun veya kentsel yerleşimin üretimiyle ilişkilendirmektedir. Bir başka Grasshopper eklentisi olan *Grasshopper Reach Analysis Toolkit* (Feng and Zhang, 2017) ise Mekan Dizimi'nin erişim analizlerini tasarım formülasyon sürecine entegre etmektedir.

Diğer araçlardan farklı olarak, SpiderWeb (Schaffranek ve Vasku, 2013) eklentisinden ilhamla geliştirilen ve ona uyumlu olarak çalışan *Syntactic* (Nourian ve diğ., 2013) ise soyut mekansal ilişkilerden olası topolojik düzlemsel yerleşimlere ulaşma çabası nedeniyle bu projenin araştırma süreci için değerli bulunmaktadır. Delft Üniversitesi'nde Pirouz Nourian ve Samaneh Rezvani tarafından geliştirilen Syntactic, Rhinoceros ve Grasshopper'da çalışan bir araç seti önermektedir (Nourian ve diğ., 2013). Syntactic, dışbükey mekanlarla çalışan etkileşimli "bubble" diyagramları oluşturmaya izin verir ve gerçek zamanlı olarak Mekan Dizimi analizlerini sunarken diyagramatik plan yerleşimleri oluşturmayı hedefler (Nourian ve diğ., 2013). Programın önerdiği mimari tasarım süreci, soyut biçimler/ilişkilenmelerin kurgulanması ile başlar. Ardından mekan büyüklüklerinin dahil edildiği etkileşimli "bubble" diyagramlar oluşturulur; coğrafik yönler hesaba katılır ve program bu sürecin sonunda biçimlenen plan şemalarını ortaya çıkarır. Aynı zamanda program, oluşan mekansal ağın, derinlik hesabını yaparak geçiş grafiklerini (justified graph) oluşturur; entegrasyon (integration), kontrol, seçim (choice) ve entropi gibi Mekan Dizimi metriklerini gerçek zamanlı olarak hesaplar. Etkileşimli bir arayüze sahip olan Syntactic, oluşan şemalarda gözlenen yan-yanalık, komşuluk, bağlılık (connectivity) ilişkilerini değerlendirerek plan yerleşimleri/örüntüleri oluşturmaktadır (Nourian ve diğ., 2013).

Sözü edilen bu yazılımlar temelde Mekan Dizimi metriklerini hesaplamak ve mimari tasarıma katkı sağlamak amacıyla geliştirilmiş olsa da her biri çeşitli araştırma alanlarına odaklanarak veya farklı yöntemler kullanarak farklı kaliteler sergilemektedir. Örneğin Axman, Segmen (Iida, 2005) ve WebMapatHome (Gil ve diğ., 2015) gibi bir kısım yazılım, Mekan Dizimi teorisinde 'mekandaki bir noktanın maksimum uzantısını temsil eden en uzun düz çizgi' (Turner ve diğ., 2005) olarak tanımlanan "eksensel çizgiler" ve bunlar aracılığıyla üretilen "segment haritalar" üzerinden çalışmaktadır. Omnivista, SPOT (Markhede ve Carranza, 2007) gibi programların dahil olduğu bir grup yazılım ise Mekan Dizimi teorisinde 'mekandaki belirli bir bakış noktasından ve belirli bir çevreye referansla görülebilen tüm noktaların bir kümesi' (Benedikt, 1979) olarak tanımlanan "izovist"ler ile hesaplama yapmaktadır. Jass (Koch, 2004), Syntactic (Nourian ve diğ., 2013) gibi bir kısım yazılım ise 'herhangi iki nokta arasında çizilen hiçbir düz çizginin o alanın dışına çıkmadığı bir alan' (Hillier ve Hanson, 1984) olarak tanımlanan

“dışbükey mekan”lar ve bunlar ile oluşturulan haritaları temel alarak çalışmaktadır. Ayrıca Confeego, DepthmapX ve Syntax2D gibi farklı yöntemleri bir arada barındıran ve daha geniş bir araştırma/uygulama alanına imkan veren bilgisayar yazılımları da bulunmaktadır.

Yukarıdaki incelemelerin yanı sıra Mekan Dizimi yazılımlarının, tasarım sürecinde ne tür bir araç olarak kullanıldığı, araştırma kapsamında yorumlanmış ve bu durum önemli bir değerlendirme alanı olarak görülmüştür. Bu kapsamda incelenen yazılımlar, analitik, dinamik ve/veya etkileşimli yapılarına göre farklılaşmaktadır. Bu alandaki öncül yazılımların, ilk Mekan Dizimi araştırmalarında olduğu gibi mevcut mekansal kurgunun analizi üzerine odaklanan, konfigürasyonel yapının anlamını çözümlemeye çalışan *analitik* ve *tanımlayıcı* bir karaktere sahip oldukları görülmektedir. Var olan yapıları veya kentsel düzenlemeleri analiz etmek/anlamak üzere çalışan ve ürettiği sayısal veri ile mekana ilişkin bir mimarlık bilgisi üretmeyi hedefleyen, böylece mekanı kavramaya / anlatmaya aracı olan yazılımlardır. Mekan Dizimi’ni yapma / eyleme anına dahil etmek yerine daha çok var olanın karakteristik özelliklerini somutlaştırarak (örneğin içinde bulunulan bağlam / kültür) tasarım sorunlarının belirlenmesine katkı sağlamak üzere kullanılan yazılımlardır. Özellikle 1988 yılında Nick Dalton tarafından geliştirilen Axman ve onu takiben üretilen *Netbox, Pesh, NewWave, SpaceBox, OrangeBox, Omnivista, Spatialist* ve daha güncel tarihli *Mindwalk, Jass, Place Syntax Tool, Confeego* gibi yazılımlar bu kapsama dahil edilebilir. Burada daha çok tasarımcının mekana ilişkin farkındalığının ve mekansal kavrayışının geliştirilmesi hedeflenmiştir. Bir sonraki adımda yine mevcut araştırma sürecine paralel olarak yazılımların tasarım sürecinde tasarlanmış kurguların olası sonuçlarını tasarımcıya sunması, tasarıma veri oluşturması hedefleriyle daha *dinamik* bir karakter kazandığı söylenebilir. Tasarlananın mekansal-sosyal bilgisi, yeni tasarımları yönlendiren veriler olarak tasarım sürecini değişime dönüşüme, yeni olasılıkları aramaya açık hale getirmektedir. Bu tür bir akılla çalışan yazılımların tasarım sürecinde ortaya çıkan ürün üzerinden aktif olarak işlem yaptığı görülürken yazılımın kullanıcısıyla (mimarla) kurduğu ilişkinin yine de tek yönlü olduğunu belirtmek mümkündür. Alasdair Turner tarafından 2001 yılında ortaya konan, daha sonra geliştirilen ve hala en etkin Mekan Dizimi araçlarından biri olarak kullanımda olan *DepthmapX ve Axwoman, Ajax, WepMapatHome, Syntax2D, Segmen, i-VALUL, sDNA, UNA Toolbox* gibi programlar bu başlık altında incelenebilir (Şekil 1).



Şekil 1. Mevcut Mekan Dizimi yazılımlarının incelenmesi.

Son zamanlarda ise tasarımcı ve bilgisayar arasında konfigürasyonel düşüncelere ilişkin karşılıklı bir diyaloğa izin veren (Nourian ve diğ., 2013), *etkileşimli (interaktif)* bir karaktere sahip yeni nesil yazılımların geliştirilmesi söz konusudur. Bunlar Mekan Dizimi'ne ait hesaplamaları tasarım sürecine dahil eden, birlikte düşünmeye ve üretmeye olanak veren araçlardır. Burada yazılım ve kullanıcı arasındaki ilişkinin çift yönlü olduğunu ve tasarım süreci ve Mekan Dizimi analizleri arasındaki ilişkinin ise sürekli birbirlerini geri besler biçime evrildiğini söylemek mümkündür. Tasarım sürecindeki her müdahale, Mekan Dizimi hesaplamalarını etkiler ve aynı zamanda analitik verinin değişimi de tasarım sürecine bir veri olarak geri döner. Kısa bir süreliğine prototip olarak kullanımda bulunan *SPOT* yazılımını ve özellikle Grasshopper eklentisi olarak çalışan *Spiderweb, Decoding Spaces Components, Syntactic, Grasshopper Reach Analysis Toolkit* gibi yazılımları bu kapsama dahil etmek mümkündür. Ancak etkileşimli olma halini her bir yazılımın nasıl yorumladığı ve çalışma sistemine nasıl entegre ettiği konusu farklılık gösterebilmektedir. Üretken ve analitik modelleri bir araya getiren ve "deneyim" odaklı bir tasarım metodolojisini kullanmaya niyet eden araştırmaların (Derix,

2014, Derix ve Izaki, 2013) bir parçası olarak düşünülebilecek bu yazılımlar tasarımcıya, tasarım sürecinde birlikte düşünebileceği, zihin açıcı araçlar sunma potansiyeline sahiptir. Mimarın tasarım sürecinde düşünme biçimini (Lawson, 2005) kendi metodolojisi içine taşıyan, tasarlayanı yaptıklarından öğrenen entelektüel bir birey (Schön, 1987), süreci problem ve çözümün bir arada geliştiği (Dorst, 2010), olasılıklara gebe bir araştırma ve keşif süreci (Dursun ve Kozikoğlu, 2017) olarak yapılandırmaya yardımcı olmaya niyet eden bu tür yazılımlar, bu araştırmanın da ilgi odağıdır.

Çalışma kapsamında Mekan Dizimi programlarını tanımlayıcı, dinamik ve etkileşimli karakterleri kapsamında değerlendirerek var olan yazılımlar arasında keskin ayrımlar üretmek yerine farklı yazılımların ve çeşitli çalışma prensiplerinin ne tür sonuçlar ortaya koyduğunu görmenin değerli olduğu düşünülmüştür. Mevcut Mekan Dizimi yazılımlarına dair yapılan incelemeler sonucunda proje kapsamında geliştirilen yazılımların ana ilgisinin bilimsel bir kavrayışın tasarım sürecine analitik, yaratıcı, dinamik ve interaktif özellikler gösterecek bir tasarım aracı olarak nasıl tercüme edileceğine ilişkin olduğu söylenebilir. Burada şu noktaların altı çizilebilir;

1. Peşinde olunan ne sadece otomatik bir form bulma, yaratma sürecine katkıda bulunacak bir araçlar setidir; ne de sadece tasarım süreci sonunda oluşan kurguların analizine ilişkin araçlar setidir. Hedef bu özellikleri bir arada bulundurabilen, mimarı ‘an’da bilgilendirerek, gerçek zamanlı bilgi ile yeni düşünceler üretimine katkıda bulunabilecek araçlar setinin araştırılmasıdır.

2. Tasarım sürecinde ana aktörün mimar olduğu düşünülmekte, bu süreç mimarin dahil olmadığı, olası form bulutu içinden seçim yaptığı bir süreç olarak düşünülmemektedir. Mimar interaktif bir karakter gösteren araçlar seti yardımıyla düşünerek bu sürece aktif katılır, bilimsel veri ile tasarımlarını değerlendirir, yeniden strüktüre eder ve tasarımı yönetir.

3. Söz konusu araçlar statik bir kurguda değil, tasarımcının bizzat dahil olabildiği, dinamik ve interaktif bir kurguda, oyun benzeri bir yapıda düşünülmektedir. *Oyun* benzeri araştırma sürecinde söz konusu bilimsel araçların tasarımcının birlikte düşünebileceği bir özellik göstermesi, bir başka deyişle sezgileri bilimsel verilerle beslemeyi hedeflemektedir.

4. Söz konusu araçların sadece belirli bir geometrik kompozisyona ulaşılması anlamında değil, hayal edilen yaşam biçiminin (kullanan bedenin/bedenlerin hareketi, bir araya gelmeler, karşılaşmalar, ayrışmalar) keşfi, değerlendirilmesi ve yeniden kurgulanması sürecine katkı yapması önemsenmektedir. “Deneyimleyen beden” ve onun eylemleri üzerinden üretilen mekansallıkların, “eskiz” benzeri bir süreçte aranması ve keşfedilmesi değerli bulunmaktadır.

Proje kapsamında yürütülen araştırmanın hedefinde; mimarlara yönelik olarak tasarlanan, bağımsız olarak kullanılabilen, Windows ve macOS ortamlarına uyumlu olarak internet üzerinde çalışabilen, Mekan Dizimi metriklerinden yararlanan, mekansal kurguların topolojik ve geometrik olasılıklarını araştıran, ağ görselleştirmesine odaklanan *oyunsallaştırılmış* bir yazılım (*SpaceChase*) yer almaktadır.

Oyun üzerine çalışmalar için Hollandalı tarihçi Johan Huizinga'nın Home Ludens (1938) adlı kitabı önemli bir referans olarak kabul edilebilir. Bu kitapta Huizinga oyunu birinin kendi için gerçekleştirdiği, mekansal ve zamansal olarak günlük yaşam pratiklerinden ayrılmış, mutlaka kendine özgü bir kural setine bağlı, serbest (ücretsiz) ve anlamlı bir aktivite olarak tanımlamaktadır (Rodriguez, 2006). Oyunun değeri ürettiği deneyimle şekillenir. Oyunun öğrenme süreçlerinde de etkin bir aktör olduğu bilinmektedir; oyuncu oyun içindeki deneyimiyle öğrenir, yaşadıklarını değerlendirebilir ve böylelikle kurduğu sebep-sonuç ilişkilerini yaşamına yansıtabilir (Yılmaz, 2016).

Çatak'a (2010) göre "oyun sadece eğlence amaçlı bir aktivite değildir; çeşitli metaforlar, kurallar ve stratejiler yoluyla duygusal gerçekliğin öznel ve kasten basitleştirilmiş bir temsilidir. Bilgisayar oyunlarının dinamik yapısı, gerçekliğin kurcalanmasını ve etkilerde bulunup tepkiler alınmasını mümkün kılmaktadır. Oyunun en önemli karakteristiği de, temsilin en bütünsel hali olan bu etkileşime izin vermesidir." (Çatak, 2010).

Oyunun dinamiklerinin yaparak öğrenme stratejisi üzerinden gelişen mimari tasarım süreci ile benzerlikler gösterdiği söylenebilir. Belirli kurallar / sınırlar içinde kullanıcıyı ana aktör yaparak etkileşim ile ilerlemesi, kullanıcıya yaptıklarından öğrenme olanağı vermesi, denemelerle ilerlemesi, gerçeği o anda simüle ederek kullanıcıyı hamlelerinin olası sonuçlarıyla yüzleştirmesi gibi sebeplerle oyun, mimari tasarım süreci için ilham vericidir. Bu çalışmaya konu olan oyunsal yazılımın da bu düşüncelerden hareket ederek mekansal deneyimi, farkındalığı zenginleştiren, etkileşim içinde yaparak öğrenme süreçlerini destekleyen, oyuncunun "birlikte düşünebileceği" bir araç olması hedeflenmiştir. Öte yandan bu oyun bilimsel veriyi de bünyesine dahil etmekte ve onu uzun soluklu bir araştırma sürecinin bir parçası haline getirmektedir.

Çalışma kapsamında Mekan Dizimi'nin ortaya koyduğu teorik yaklaşım ve araç setinden yararlanarak geliştirilen oyunsal yazılıma "SpaceChase" adı verilmiştir. SpaceChase mekansal parçaları ve onlara eşlik eden bağlama ait verileri düğümler aracılığıyla temsil ederek ve aralarındaki ilişkileri kurup bozarak olası bir araya gelmeleri, konfigürasyonları, yerleşimleri araştırmaya niyet etmektedir. Yapı ölçeğinde çalışmak üzere tasarlanan yazılım bugüne dek

konut, kreş ve restoran gibi üç tipoloji için üretilen senaryolar kapsamında çalışılarak geliştirilmiştir. Bilimsel veriler üreten bu oyunsal arayüzün mimarlar dışında geniş bir kullanıcı profili ile de etkileşime geçebileceği düşünülmüştür. Kullanıcı kurup bozduğu mekansal ağlarla belirli kaliteler ararken ve çeşitli olasılıkları denerken arka planda tüm olasılıklar ilgili Mekan Dizimi metrikleri ile değerlendirilebilir ve ölçülebilir hale gelmektedir. Oyunsal arayüz ön planda görünür iken, mekana ilişkin sentaktik veriler daha bilimsel bir bakış gerektirdiği için istenildiğinde ulaşılabilir nitelikte tasarlanmıştır.

Bu yazılıma yönelik araştırma kapsamında tasarım sürecinin karar alma / karar değiştirme yapboz ortamına paralel olarak, bağlam bileşenlerinin (bağlama ilişkin mekansallıklar) görece statik, tasarlanan bileşenlerin (tasarıma ilişkin mekansal birimler) hem aralarında hem de bağlam bileşenleri ile ilişkilerinde değişken/ dinamik olduğu ana fikrinden yola çıkılmıştır. Bu ilişkilerin kümelenme ve yeni ilişkileri doğurma kapasitesi önemsenmiştir. Ayrıca tasarımcının bileşenleri tarif edeceği, dilediğinde yok edeceği ya da ekleme yapacağı öngörülmüştür. Tasarımın son ürününe değil sürecine olan bu odak, bu oyunsallaştırılmış yazılımın sözü edilen mevcut yazılımlardan farklılığının ana eksenidir. Bağlamın ve programın tasarımcı tarafından yorumu böylece devreye girmektedir. Tasarımcının yorumunun tasarım sürecinin başında veriye bağlı ve sezgisel yönleri, süreç içinde ilişkisel konumlanmalar sonucunda oluşan veri ile desteklenmektedir. Bu ilişkiler seti karmaşıktır, etkileşimle değişebilen karakterdedir. Amaç, tasarlanan arayüzle bu kompleks yapının okunaklı, görünür ve tartışılabilir olabilmesidir.

Proje kapsamında yürütülen ve bu oyunsal yazılıma bir ön hazırlık olan bir diğer çalışma da Rhinoceros/ Grasshopper yazılımı ile birlikte çalışan bir eklentinin (*SpaceChase for Grasshopper*) geliştirilmesidir. Paralel olarak yürütülen bu iki çalışma, süreç boyunca birbirini beslemiş, araştırmayı farklı araç setleriyle zenginleştirmiştir. SpaceChase for Grasshopper, oyunsal yazılımın üretim sürecine destek olmak üzere, atölyeler aracılığıyla mimarlık öğrencilerinden geri dönüş almayı kolaylaştırmak, farklı taslak senaryolar üzerinde çalışmayı hızlandırmak, dinamik ağ yapısı ve ilgili Mekan Dizimi analizlerinin etkileşimini görerek test edebilmek gibi hedeflerle geliştirilmiştir. SpaceChase for Grasshopper'ın tasarımı, özellikle Spiderweb (Schaffranek ve Vasku, 2013) ve Syntactic (Nourian ve diğ., 2013) yazılımlarının kuramsal çerçevelerini ve çalışma prensiplerini inceleyerek, onların eksik kaldığı alanlara odaklanarak başlamıştır.

Mekan Dizimi hesaplamalarını analitik olarak sağlayan ve dinamik ağ yapısı aracılığıyla kullanıcı etkileşimini hedefleyen yazılım, incelenen Mekan Dizimi yazılımlarından farklılaşmakta ve bu literatürde yeni tartışmalara olanak vermektedir. En büyük farklılık,

geliştirilen yazılımın, bugüne dek tasarlanan ve statik ölçümlere dayanan yazılımların aksine, bu ölçümler ile dinamik bir ağ elde eden, hazır grafikler üretmek yerine CAD ortamında kullanıma hazır (nokta, çizgi, düzenli ve sıralı veri gibi) çıktılar sağlayan ve bunu tasarım sürecinde kullanıcının yaratıcı kullanımına bırakan bir yazılım olmasıdır. Ayıca tasarım sürecine dahil ettiği tasarlanan konfigürasyonun bulunduğu bağlama ait veriler sayesinde çevresi ile bütünleşik olarak tasarım yapmaya ve eş zamanlı olarak tasarlananı analiz etmeye olanak sağlamaktadır. Araştırma kapsamında geliştirilen yazılım, yukarıda sözü edilen birçok programın yalnızca sayısal olarak verdiği bazı ilişkileri/durumları grafik olarak ve eş zamanlı bir biçimde görebilmeyi sağlamakta, sayısal verilerin ardındaki aklı dinamik ağ yapısı aracılığıyla görünür kılmaktadır. Konfigürasyon içindeki her mekan sentaktik değerleri bağlamında (örneğin, derinlik ve entegrasyon) hareketlenir, birbirini iter ve/ya çeker. Dinamik olarak bir etki alanı ve mekansal gruplaşmalar oluşturur. Örneğin, Mekan Dizimi teorisine göre “bir mekan ne kadar entegre ise sistemdeki diğer tüm mekanları yakınına çekecek, ne kadar ayrı ise diğer tüm mekanları kendinden uzaklaştıracak” (Hanson, 2003) biçimde çalışır; bu durum, arayüzde görünen entegrasyon değerine ait sayısal verinin yanı sıra oluşan dinamik ağ yapısının hareketi üzerinden de entegrasyon bilgisini okumayı mümkün kılmaktadır. Bunun yanı sıra, tasarım sürecinde tasarlayanın birlikte düşünmesini sağlayan dinamik ağ, kümelenme katsayısı yüksek olan mekansal birimleri bir araya getirerek olası yan-yanalıkları görsel olarak da kullanıcıyla paylaşır. Sayısal verilerin ve mekansal ölçümlerin görsellenebilir olması ve bunların tasarımcı tarafından anlık olarak izlenebilir olması sayesinde mimari tasarım sürecine Mekan Dizimi hesaplamaları doğrudan katkı sağlayabilmektedir.

3. GEREÇ VE YÖNTEM

Projenin ana hedefi, Mekan Dizimi metriklerinden yararlanan, mekansal kurguların topolojik ve geometrik olasılıklarını araştıran, *oyunsallaştırılmış* bir yazılımın gerçekleştirilmesidir. Bu doğrultuda kuramsal alt yapının güçlendirilmesine öncelik verilmiş, ardından yazılım kodlama çalışmalarına odaklanılmış ve geliştirilen yazılım/lar test edilerek güncellenmiş, tartışmaya açılmıştır. Proje sürecinin dört ana adımda kurgulandığı söylenebilir:

- **Mevcut araştırma birikiminin dokümantasyonu**. Bu adım tasarım sürecinin doğasını, morfolojik çalışmaların, mimarlıkta ağ düşüncesinin, grafik-teorik araçların ve özellikle Mekan Dizimi'nin tasarım sürecine katkısını irdeleyen kavramsal, teorik alt yapının kurgulanması çalışmalarını içermektedir. Bu adımda Mekan Dizimi odaklı mevcut bilgisayar yazılımlarının olanak ve sınırlamalarının irdelenmiş, bu doğrultuda geliştirilen yazılımın sahip olması beklenen kalitelere vurgu yapılmıştır (Bkz. Bölüm 2.).
- **Yazılım/ların tasarımı.** Bu adım bilimsel bilgiyi tasarım sürecine taşımaya niyetli bir "tasarım araştırma aracı"nın geliştirilmesine odaklanmıştır. Bu araç mimarlıktaki deneyerek yapma, yaptıklarından öğrenme pratiğine benzer bir işleyiş mekaniğine sahip olan "oyunsallaştırılmış bir bilgisayar yazılımı"dır. Bu adım, Mekan Dizimi'nin teori ve araç setinden yararlanan söz konusu yazılım/lara ilişkin kodlama, oyun akış ve senaryolarının kurgulanması ve grafik arayüz geliştirme çalışmalarını içermektedir (Bkz. Bölüm 4.1 ve 4.2).
- **Yazılım testleri ve geri bildirimler.** Bu adım hem araştırma grubunun kendi içinde gerçekleştirdiği testlerle hem de düzenlenen atölye çalışmalarıyla söz konusu yazılım/ların potansiyellerinin ve sınırlamalarının değerlendirilmesine odaklanmıştır. Atölyeler farklı okul ve sınıflardan lisans ve lisansüstü eğitimi alan mimarlık öğrencileriyle ulusal ve eCAADe 2020 kapsamında ise uluslararası bir platformda gerçekleştirilmiştir. Bu aşamada toplanan veriler, öğrencilerin yazılımı kullanırken ürettiği mekansal ilişki ağlarını, bunların analiz sonuçlarını ve kullanıcıların bu sonuçlar üzerinden yaptıkları yorumları içermektedir. Fiziksel ortamda gerçekleştirilen bu etkinlikler yaşanan gelişmeler sonucunda dijital ortama taşınmıştır. Değerlendirmelere ilişkin veriler mülakat, gözlem, video-ses kaydı teknikleri ve yapılan anket çalışmaları ile toplanmış ve analiz edilmiştir (Bkz. Bölüm 4.3.1). Böylelikle test edilen yazılımın kullanım şekli, içeriği ve kapasitesi ile ilgili elde edilen veriler nicel olarak yorumlanmıştır. Atölye üretimleri üzerinden mekansal ilişkilenme ve planimetrik kurgu arasındaki korelasyon araştırılmış, alınan geri bildirimler üzerinden yazılım/ların güncellenmesi yapılmıştır.
- **Bilgi paylaşımı ve değerlendirmeler.** Bu adımda ortaya çıkan bilgi birikimi, deneyim ve üretimler düzenlenen bilimsel toplantılar, tasarlanan web sayfası ve hazırlanan bilimsel yazı ile ulusal ve uluslararası çeşitli ortamlarda paylaşılmıştır. Dijital ortamda birer günlük etkinlikler

olarak planlanan bilimsel toplantılarda Mekan Dizimi araştırmalarına odaklanan ve “oyun” üzerine çalışan akademisyenlerle ulusal ve uluslararası düzeyde bir tartışma ortamı yaratılmış, alınan geri bildirimler üzerinden oyunsal yazılımın değerlendirilmesi yapılarak gelecek çalışmalar için hedefler kurgulanmıştır (Bkz. Bölüm 4.3.2 ve 4.3.3).

4. BULGULAR

4.1 SpaceChase for Grasshopper

Proje kapsamında geliştirilen "*SpaceChase for Grasshopper*", tasarımcıların ve mimarların dinamik ve etkileşimli ağ yapıları kurmalarına ve analiz etmelerine olanak sağlayan bir yazılım eklentisidir. Yazılım eklentisi *Oğulcan Üneşi* tarafından geliştirilmiştir.

SpaceChase for Grasshopper, Rhinoceros yazılımının algoritmik modelleme arayüzü olan Grasshopper içerisinde çalışan bir eklentidir. Rhinoceros, mimarların ve tasarımcıların üç boyutlu modeller oluşturmasını, analiz etmesini, görselleştirmesini ve dijital üretim yöntemleri kullanarak gerçekleştirmesini kolaylaştıracak şekilde geliştirilmiş bir bilgisayar destekli tasarım (BDT) yazılımıdır. Bu yazılım üzerinde çalışan Grasshopper arayüzü, tasarımcıların gelişmiş kodlama yeteneklerine ihtiyaç olmadan algoritmik biçim araştırmaları yapmasını mümkün kılmaktadır. Grasshopper, Rhinoceros 6 sürümünde bir eklenti olmaktan çıkarak yazılımın ana bileşenlerinden birisi haline getirilmiştir. Veri akışı modellemesi (*dataflow modeling*) kodlama paradigmasını temel alan Grasshopper arayüzü, kullanıcıların ekrandaki kanvas üzerine bileşenleri (*component*) yerleştirdiği ve bu bileşenleri birbirine bağladığı bir yönlü diyagram sistemiyle çalışır. Bileşenler girdi (*input*) ve çıktı (*output*) parametrelerine sahiptir. Genel olarak; sayı, alfanümerik karakter veya koordinat gibi girdiler bileşenler tarafından nokta, çizgi ve yüzey gibi geometrik objelere dönüştürülmektedir. Arka arkaya bağlanan bileşenler, bir zincir ve ağ yapısı meydana getirmekte ve geometrik modelleme sürecinin algoritmik ifadesini oluşturmaktadır. Tasarımcılar bu algoritmik yapının adımlarına ve girdilerine müdahale ederek tasarımlarında hızlıca revizyon ve türetme yapabilmektedir. Grasshopper bir topluluk projesi olarak gelişmektedir. Geometrik modelleme sürecini hızlandırmasına ek olarak, bu topluluk tarafından geliştirilen yeni bileşenlerle yeni özellikler kazanmaktadır.

4.1.1 Hedefler / Adımlar

Projenin üst çerçevesi içerisindeki bu yazılım eklentisinin alt amaçları aşağıdaki gibidir:

- Mekan Dizimi'nin teori ve yöntemleriyle uyumlu ağ yapılarının, mimar ve tasarımcıların aşina olduğu bir bilgisayar destekli tasarım (BDT) ortamında oluşturulması,

- Oluşturulan ağ yapılarının dinamik ve etkileşimli hale getirilmesi, hareket biçimlerinin gözlemlenmesi ve analiz edilmesi,

- Hızlı ön prototiplerin üretilmesiyle oyunsallaştırılmış yazılımın geliştirme sürecine katkıda bulunulması.

SpaceChase for Grasshopper geliştirme süreci üç alt iş paketi (adım) tanımlamıştır:

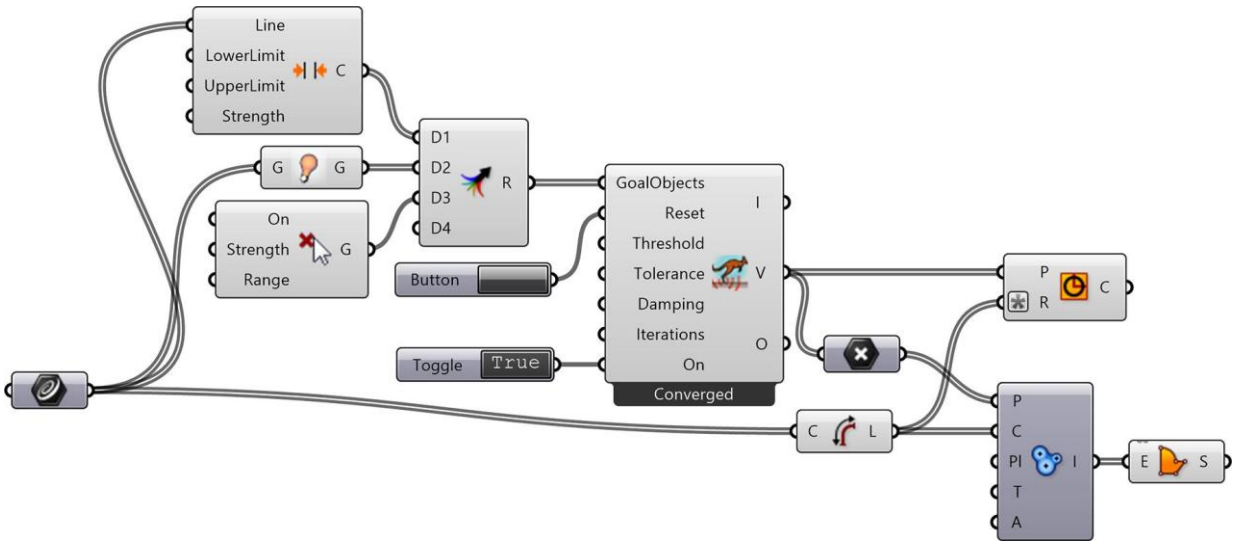
- **Ağ yapılarının Grasshopper içerisinde kodlanması:** Bu alt iş paketi, mekanların ve mekanlar arasındaki ilişkilerin dijital ortamda nasıl bir teknikle kodlanacağını ve temsil edileceğini belirlediği için geliştirme sürecinin temel öğelerinden birisidir. Yapılan ilk prototipte, mevcut kodların kullanımının ileri aşamalarda sınırlayıcı olacağı öngörülerek özgün bir eklentinin geliştirilmesine gerek görülmüştür. Aşağıdaki bölümlerde geliştirme sürecinde araştırılan ve denenen farklı yaklaşımlar açıklanacaktır.

- **Ağ yapılarının dinamik ve etkileşimli hale getirilmesi:** Bu amaçla Grasshopper'ın mevcut bileşen setlerinden birisi olan *Kangaroo* kullanılmıştır. Bir geometrik optimizasyon paketi olan Kangaroo, verilen geometrik hedefler (*goals*) yardımıyla çalışmaktadır. Bu geometrik hedefler ekran üzerinde optimize edilirken, gerçek zamanlı ve kullanıcı etkileşimine açık bir animasyon olanağı sunmaktadır. Dolayısıyla, SpaceChase for Grasshopper'ın ihtiyaç duyduğu etkileşimli ve dinamik keşif ortamını mümkün kılmaktadır. Bu alt iş paketi, mevcut Mekan Dizimi yazılımlarında ve eklentisinde bulunmayan etkileşim ve dinamik graf yapısını özgün bir karakter olarak ortaya çıkarmaktadır. Eklentinin ileri sürümlerinde Kangaroo bileşenlerinin sistemden çıkartılarak, onların yerine orijinal bir geometrik optimizasyon altyapısının kodlanması hedeflenmiştir. Fakat yapılan ön çalışmalar bunun süre ve teknik olanaklar açısından makul olmayacağını gösterdiği için mevcut sürümlerde Kangaroo altyapısı kullanılmaya devam edilmiştir.

- **Mekan Dizimi metriklerinin hesaplanması:** Bu metrikler ilk alt iş paketinde kodlanan ağ altyapısı üzerinde bazı hesaplamalar yapmayı gerektirmiştir. Metriklerin ikinci alt iş paketinde tanımlanan dinamik ve etkileşimli ortamla olan ilişkisi ise geliştirme süreci boyunca tartışılan ve üzerinde çalışılan bir konu olmuştur. Dolayısıyla bu alt iş paketinin geliştirilmesi, diğerlerine kıyasla daha geç bir adımdır. Ön çalışmalarda ve ilk prototiplerde bu nedenle Mekan Dizimi metrikleri üzerinde çalışmak mümkün olamamıştır. SpaceChase for Grasshopper'ın mevcut sürümü (0.3.3); ağ yapıları üzerinde Dijkstra'nın (1959) en kısa yol (*shortest path*) hesaplamasından hareketle çeşitli Mekan Dizimi metriklerini hesaplayabilmektedir. 0.3.3 sürümünde bu metrikler bağlılık (*connectivity)*, ortalama derinlik (*mean depth)*, seçim (*choice)*, kontrol (*control)*, entegrasyon (*relative asymmetry*, *real relative asymmetry)* dur. Ayrıca seçili bir noktadan mekansal ilişki grafikleri (*justified graph)* yazılım tarafından otomatik olarak çizilebilmektedir. Bu özellikler bir sonraki bölümde açıklanacaktır.

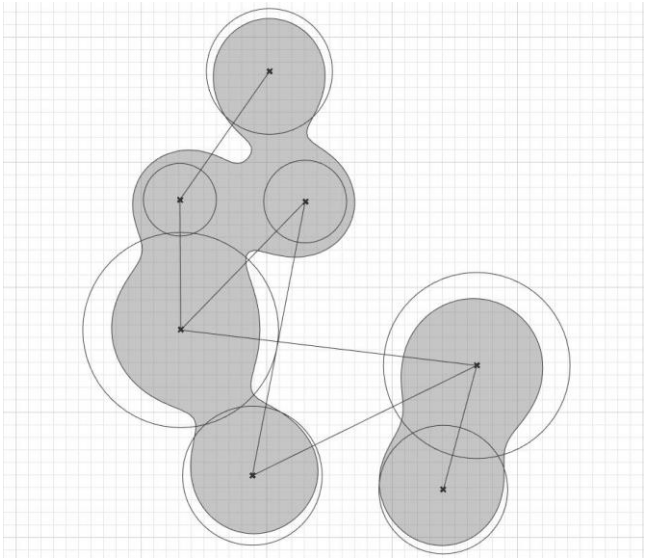
4.1.2 Geliştirme Süreci

Yukarıda özetlenen alt iş paketleri değerlendirildiğinde; SpaceChase for Grasshopper'ın geliştirme stratejisinin iki ana aşamadan oluşması gerektiğine karar verilmiştir. İlk aşama; Grasshopper'ın mevcut bileşenlerinden faydalanılarak yukarıda özetlenen alt iş paketlerinin taslaklarının oluşturulmasıdır. Bu aşamanın hedefi henüz bir eklenti geliştirmeden önce, dinamik ve etkileşimli ağ yapılarının potansiyellerinin mevcut olanaklarla modellenip test edilmesidir ve teknik gereksinimlerin tespit edilmesidir. Bu amaçla çok sayıda Grasshopper kodu oluşturulmuştur. Bu kodlar genel olarak ağ yapılarının Grasshopper üzerinde hazır bileşenler yardımıyla modellenmesi ve Kangaroo bileşenleri yardımıyla da dinamik ve etkileşimli bir şekilde sunulmasını içermiştir. Bu amaçla gerçekleştirilen çalışmalarda dört temel kilometre taşı tespit edilmiştir. Aşağıdaki bölümde bu dört kod (prototip) açıklanmıştır.



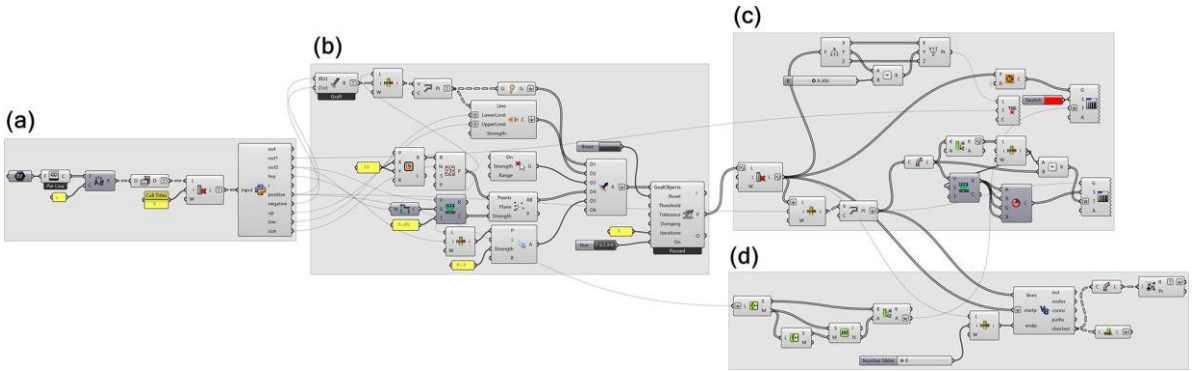
Şekil 2. SpaceChase for Grasshopper Work-in-progress (WIP) Prototip #1 (Ekim 2019). Veri akış diyagramı.

Şekil 2'de sunulan Grasshopper kodu, eklentiden beklenen dinamik ve etkileşimli ağ altyapısının potansiyellerinin ve teknik altyapısının araştırıldığı ilk denemedir. Bu prototip henüz Mekan Dizimi metrikleri ile ilgili hesaplamalar içermemektedir. Kullanıcılar, Rhinoceros ekranında oluşturdukları çizgilerle basit bir ağ yapısı oluşturmakta ve fare yardımıyla bu yapıya hareket katabilmektedir (Şekil 3).



Şekil 3. SpaceChase for Grasshopper WIP Prototip #1. Ekran görüntüsü. Kullanıcı, düğüm noktalarını fare yardımıyla sürükleyip itme ve çekme hareketlerini tetikleyebilmektedir.

Bu prototipte hareket Kangaroo'nun temel bileşenleri tarafından belirlenen, noktalar arasındaki çekme ve itme güçleri tarafından oluşturulmaktadır. Bu itme ve çekme güçleri Rhinoceros ekranına çizilen bağlantı çizgilerinin uzunluklarından alınmaktadır. *Clamp Length* isimli Kangaroo *Goal* nesnesi kullanılarak bağlantı uzunluklarının belirli toleranslar içerisinde kalmaları sağlanmaktadır. Sistemin hareketi esnasında noktalar arası mesafe o noktaları birleştiren çizginin tolerans sınırından fazla ise Kangaroo bu noktaları birbirine çekme yönünde tepki vermekte, tersi durumda ise itme yönünde çalışmaktadır. Böylelikle ağ yapısı küçük bir müdahale ile devinime girebilmektedir. Son adımda hareketli noktalar Grasshopper'ın *Metaball* bileşenleri tarafından iki boyutlu akışkan yüzeylere dönüştürülerek görselleştirilmektedir. Bu prototipte ağ yapısına müdahale edilmezse bir süre sonra sistem Kangaroo tarafından optimize olarak hareketine son vermektedir. Sistemin devinim halinde kalması beklendiği için optimizasyonun engellenmesi gerektiği anlaşılmıştır.



Şekil 4. "SpaceChase for Grasshopper WIP" Prototip #2 (Aralık 2019). Veri akış diyagramı. Bu diyagram daha detaylı olarak EK 1'de verilmiştir.

Prototip #1'den sonraki çalışmaların veri akış diyagramları gruplar halinde düzenlenmiştir. Şekil 4a'da görülen grup, ağ yapısında kullanılacak düğüm ve bağlantı verisini .csv formatındaki bir tablodan okuyarak sisteme aktarmaktadır. Bu işlev, Prototip #2'de çalışmaya eklenen bir yeniliktir.

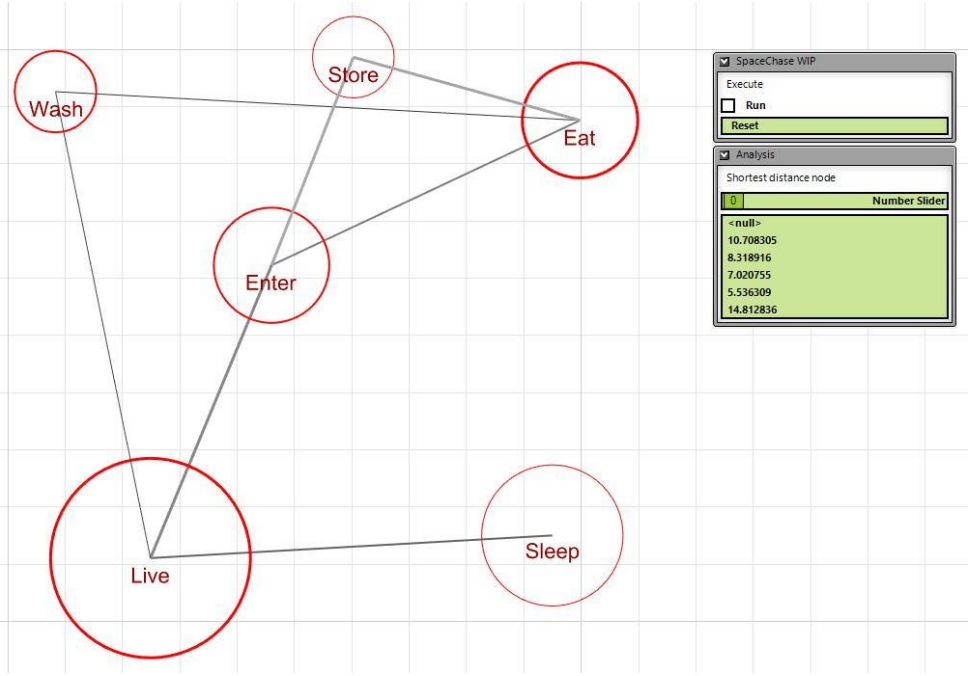
Tablo 1. SpaceChase for Grasshopper prototiplerinde kullanılan örnek .csv tablosu.

m2	title	weight	positive relations		metrics		negative relations		metrics
30	Live	0	Live	Eat	0	10	Live	Sleep	5
10	Eat	5	Live	Wash	5	30	Eat	Sleep	15
5	Wash	0	Live	Sleep	20	30			
15	Sleep	3	Live	Enter	0	5			
10	Enter	10	Eat	Wash	0	10			
5	Store	5	Eat	Enter	5	20			
			Eat	Store	0	10			
			Wash	Sleep	0	5			
			Enter	Store	0	10			

Tablo 1'de Prototip #2'de kullanılan .csv tablosu görülmektedir. Tablonun ilk üç sütununda düğüm noktaları ile ilgili bazı bilgiler tutulmaktadır. Sonraki dört sütun pozitif (birbirini çeken) ilişkileri, en son üç sütun ise negatif (birbirini iten) ilişkileri modellemektedir. Girilen bu ilk veriler test amaçlı olup, herhangi bir eğitsel programa dayanmamaktadır. Bu veriler Şekil 4a'daki kod vasıtasıyla Grasshopper'a aktarılmakta, Şekil 4b'deki grupta ise dinamik ve etkileşimli bir ağ yapısına dönüştürülmek üzere Kangaroo optimizasyon hedeflerine (*goal*) dönüştürülmektedir. Bu prototipte kullanılan üç tür hedef vardır (Şekil 4b ve EK 1). *Rigid Point Set*, verilen noktanın bağıl pozisyonunu korumasını sağlamaktadır. Düğüm noktalarının ağırlık (*weight*) değerleri ve koordinatları bu bileşene girilerek her mekanın simülasyondaki ağırlığı ve merkezi olma hali kontrol edilebilmektedir. Bu prototipte düğüm noktalarının başlangıç koordinatlarının rastgele bir biçimde üretilmesi sağlanmış, böylelikle her simülasyonda farklı başlangıç konfigürasyonlarının yol açtığı sonuçlar test edilmiştir. Bir diğer hedef olan *Anchor Point*, bazı noktaların pozisyonlarını sabitler ve hareket etmemelerini sağlar. Birinci prototiple yapılan denemelerde hareketli ağ yapısının kolaylıkla kullanıcı kontrolünden çıkarak ekran dışına savrulduğu gözlemlendiği için, bu tür sabitleyici ağ bileşenlerinin olması gerektiğine karar verilmiştir. İleride bağlamsal düğümler (*contextual node*) biçiminde tariflenecek bu yapılar Prototip #2'de henüz sadece hareketsiz düğüm noktaları olarak tanımlanabilmiştir. Prototip #2'nin diğer bir hedefi ise Prototip #1'de de kullanılan *Clamp Length*'tir ve görevi verilen bağlantı uzunluklarını belirli toleranslar içerisinde sabitlemektir. Böylece, kullanıcı ekranda düğüm noktalarını sürükleyerek

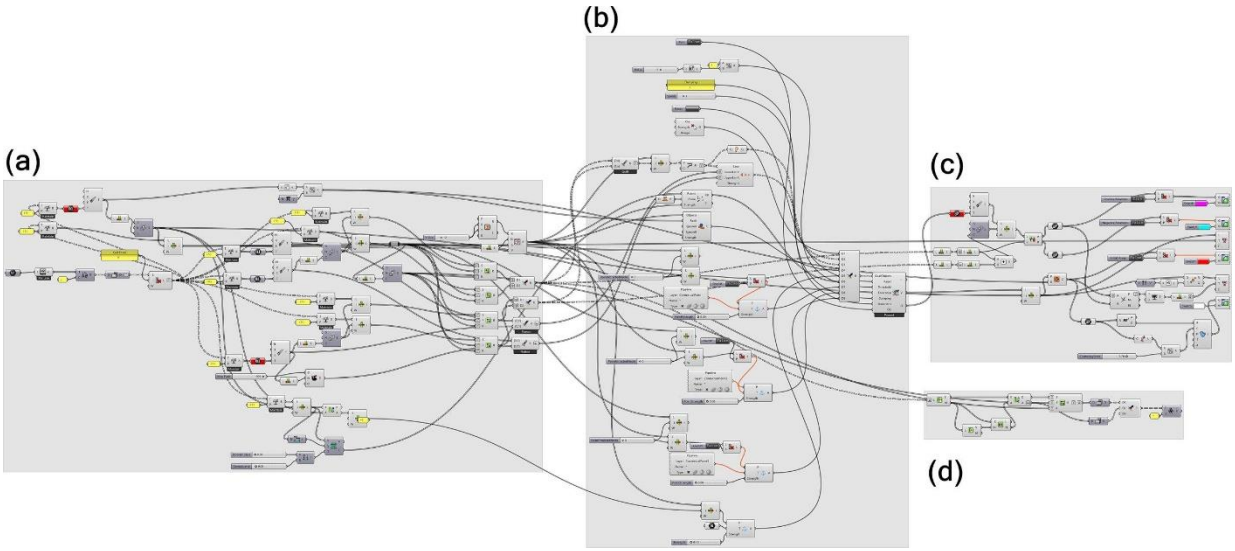
hareket ettirdiğinde yukarıda ifade edilen üç hedef senkronize bir biçimde çalışarak ağ simülasyonunu oluşturmaktadır. Şekil 4-c'de görülen grup, Grasshopper'ın bileşenlerini kullanarak ağ yapısındaki çember, nokta ve çizgileri görselleştirmektedir. Şekil 4-d'de görülen grup ise Prototip #2'nin çalışmaya kattığı bir başka yeniliktir. Bu gelişme ile beraber ilk defa dinamik ağ yapısı üzerinde Mekan Dizimi metriklerinin hesaplanmasına yönelik çalışmalar başlamıştır. Mekan Dizimi metriklerinin bir çoğunun hesaplama yönteminin temelinde bulunan, Dijkstra'nın ağ üzerinde en kısa yolu bulma algoritması (*shortest path*), bir VB.NET kodu olarak sisteme entegre edilmiştir. Böylelikle, oluşturulan ağ yapılarında her düğüm noktasının pozitif bağlantı sayısı ile, iki düğüm arasındaki en az adımlı yol ve en kısa toplam mesafe hesaplanabilmektedir.

Bu prototiple beraber, Grasshopper'ın arayüz oluşturma özelliği (*remote panel*) kullanılmaya başlanmıştır. Böylelikle Şekil 5'te görülen çalışma düzeni oluşturulmuştur. Kullanıcılar, .csv formatında hazırladıkları ağ modelini sisteme yükledikten sonra Rhinoceros ekranında düğüm noktalarını sürükleyerek hareket ettirebilmekte, açılan panelde ise simülasyonu kontrol etmekte ve seçtikleri bir düğüm noktasından diğer düğüm noktalarına olan en kısa yolları gerçek zamanlı olarak görüntüleyebilmektedir. Bu prototipte ağ yapısı simülasyonda önceden tanımlanmaktadır. Düğüm noktalarını ekran üzerinde sürüklemek ağ topolojisini değiştirmeyeceği için yapılan en kısa yol hesaplamaları da sabittir. Mekan Dizimi metriklerinin dinamik ağa nasıl bir etkide bulunabileceği konusu tartışılmaya devam edilmiştir.



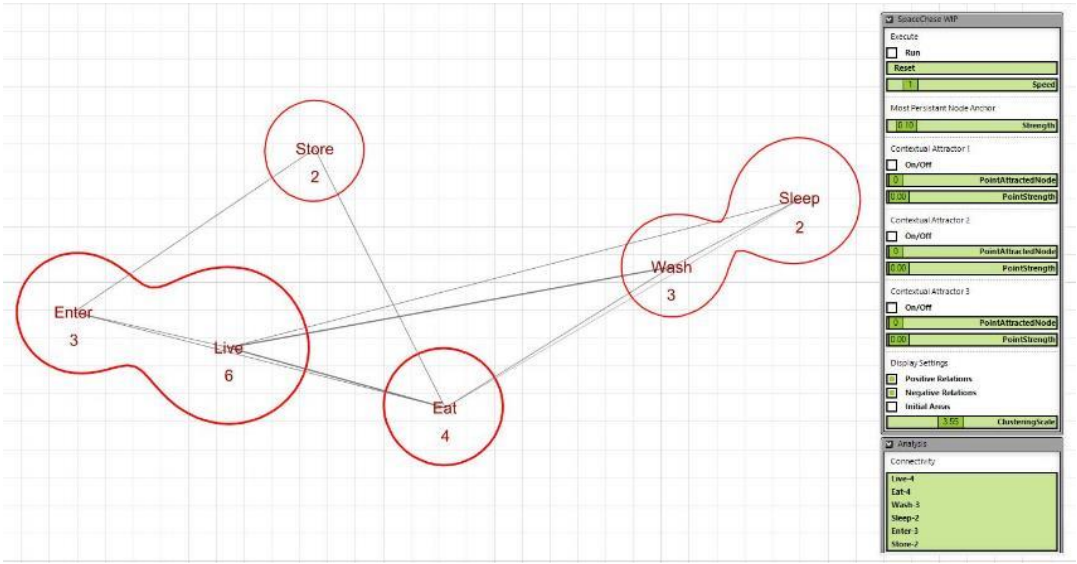
Şekil 5. "SpaceChase for Grasshopper WIP" Prototip #2 (Aralık 2019). Ekran görüntüsü.

Prototip #2'nin ürettiği ağ yapıları, hareketin başlamasından belirli bir süre sonra eğer kullanıcı tarafından harekete geçirilmezse optimize olarak durabilmektedir. Bu aşamada hızlanan geliştirmelerin sonunda her üç alt iş paketini de içeren Prototip #3 oluşturulmuştur.



Şekil 6. "SpaceChase for Grasshopper WIP" Prototip #3 (Aralık 2019). Veri akış diyagramı. Bu diyagram daha detaylı olarak EK 2'de verilmiştir.

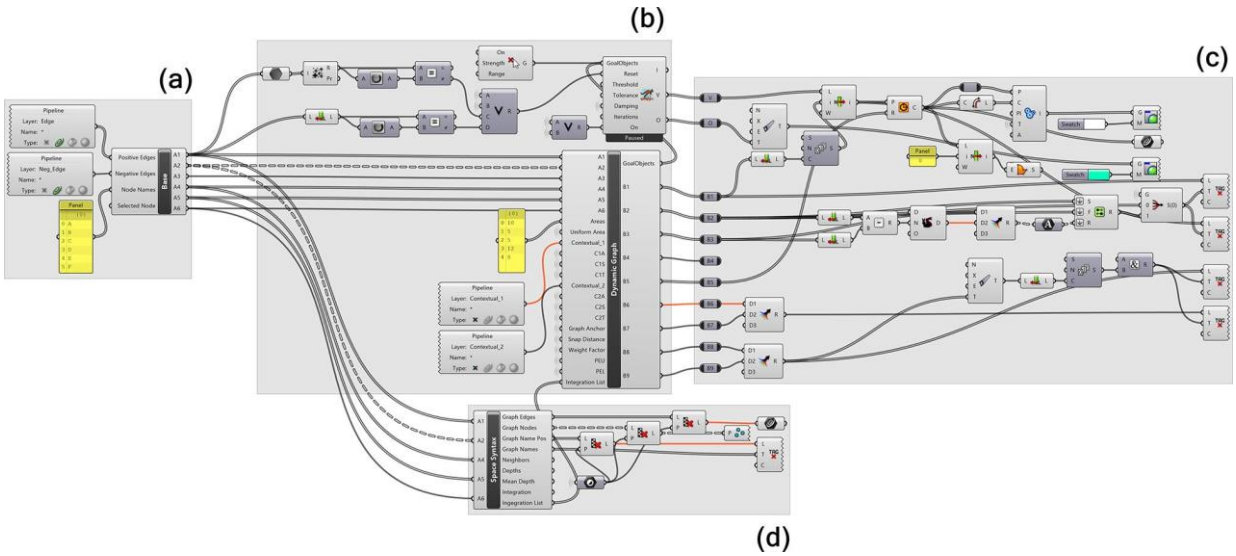
Bu kod, csv tablo formatında kaydedilen mekansal ilişkileri okuyarak ağ yapısı haline getirmekte ve bir önceki prototipin aksine sürekli devinim halinde kalacak şekilde sunmaktadır. Bu prototipte ayrıca bağlamsal düğüm (*contextual node*) ismi verilen ve tasarımcının bağlam ile ilişki kurmasını hedefleyen durağan noktalar tanımlanabilmektedir. Her düğüm noktası için bağlılık (*connectivity*) değerleri hesaplanabilmektedir. Şekil 6-a'da görülen grup, .csv uzantılı bir tablo dosyasını okuyarak SpaceChase için uygun formata getirilmesini sağlamaktadır.



Şekil 7. "SpaceChase for Grasshopper WIP" Prototip #3. Ekran görüntüsü.

Şekil 6-b, Kangaroo optimizasyonu için gereken ön hazırlıkların yapılması ve dinamik ağ yapısının oluşturulmasını sağlamaktadır. Bu grupta yapılan düzenlemeler ile animasyonun sürekliliği elde edilmiştir. Ayrıca bu düzenlemeler bağlamsal düğümlerin Rhinoceros katmanlarında bulunan geometrik noktalar üzerinden otomatik olarak Grasshopper'a aktarılmasını sağlamaktadır. Geliştirilen bu yöntemler ileri aşamalarda da kullanılacaktır. Şekil 6-c grubu, dinamik ağ yapısının Rhinoceros ekranında görselleştirilmesi için bazı yeni ayarlamaları olanaklı kılmaktadır. Şekil 6-d grubu ise düğümlerin ilişki sayılarını hesaplayarak Grasshopper arayüzünde sunmaktadır. Bu hesaplamalar ileri aşamalarda daha detaylı biçimde ele alınacak olan Mekan Dizimi metrikleri için bir ön hazırlık niteliğindedir.

Prototip #2 ve #3 arasında bulunan bir sürüm *Education and Research in Computer-aided Architectural Design in Europe* (ECAADE) 2020 konferansında bildiri olarak sunulmuştur. Prototip #3, 5-12 Aralık 2019 tarihlerinde İTÜ Mimarlık Fakültesi'nde düzenlenen birinci öğrenci atölyesinde test edilmiştir. Testler sonucunda kodun nasıl geliştirilebileceği ve ileride eklentinin nasıl programlanması gerektiği konularında geri bildirimler elde edilmiştir. Bu atölye ile ilgili detaylı açıklamalar 4.3 numaralı bölümde sunulmuştur. Öğrenci atölyesine paralel olarak projenin diğer çıktısı olan oyun yazılımı ile ilgili çalışmalar da bu dönemde başlamıştır. Grasshopper'da yapılan ve yukarıda açıklanan prototipler oyun yazılımının gelişim sürecinde de birer ön çalışma olarak kullanılmıştır. Böylelikle dördüncü ve son kilometre taşı olan Grasshopper kodu oluşturulmuştur (Şekil 8).

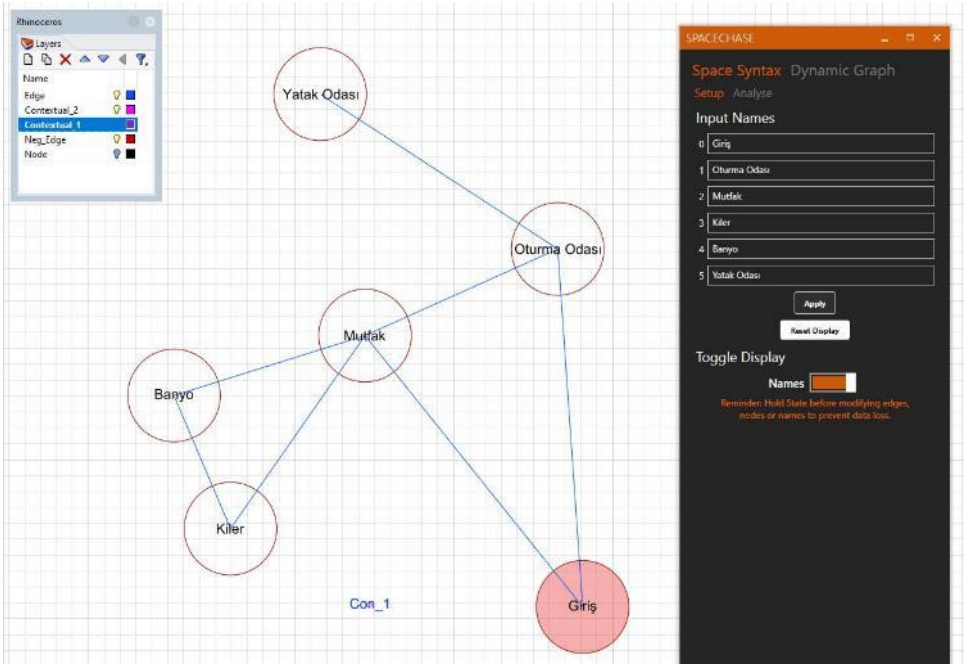


Şekil 8. "SpaceChase for Grasshopper WIP" (Haziran 2020). Prototip #4; veri akış diyagramı. Bu diyagram daha detaylı olarak EK 3'te verilmiştir.

Dördüncü prototip Şekil 8'te gösterildiği gibi, bir önceki prototipe kıyasla daha düzenli ve sistematik bir yapıya sahiptir. Karmaşık hesaplamaların yapıldığı parçalar birer küme (*cluster*) olarak gruplanmakta, böylece hem kodun daha sade ve sistematik olması sağlanmakta, hem de şifreleme yapılarak kodun özgünlüğü korunmaktadır.

Bu prototip, ağ yapısını Rhinoceros ekranındaki çizgilerden almaktadır. Dolayısıyla bir önceki prototipe kıyasla daha esnek bir kullanım şekline sahiptir. Kullanıcılar ağ yapısında değişiklik yapmak istediklerinde bir .csv tablosuna dönmek yerine doğrudan Rhinoceros'taki nokta ve çizgilere müdahale ederek bunu yapabilmektedirler. Şekil 8-a'da görülen küme bu işlevin kontrolünden sorumludur. Elde edilen ağ verisi, Şekil 8'de görüldüğü gibi iki ayrı kümeye yönlendirilmektedir. Bunlardan ilki, Şekil 8-b'de görülen *Dynamic Graph* isimli bileşen kümesidir. Bu küme, önceki prototiplerde olduğu gibi, ağ yapısı ile Kangaroo simülasyonu arasında veri düzenleyici görevi görmektedir. Bu küme, ağ verisini ve bağlamsal düğümler, mekanların alan bilgisi gibi ek verileri de toplayarak Kangaroo bileşeninin kullanabileceği geometrik hedeflere dönüştürmektedir. Şekil 8-c'de görülen bileşenler Kangaroo'dan elde edilen dinamik ağı Rhinoceros'ta görselleştirmekten sorumludur. Ağ verisinin kullanıldığı ikinci küme olan *Space Syntax*, girdi olarak verilen mekansal ağın derinlik, ortalama derinlik ve entegrasyon değerlerini hesaplamaktadır.

Prototip #4'ün çalışmaya iki temel ve özgün yenilik kattığı görülmüştür. Bunlardan ilki, Mekan Dizimi metriklerinin Grasshopper ortamında önceki çalışmalara göre daha etkin biçimde hesaplanmaya başlanmasıdır. Araştırmanın bu aşamasında elde edilen deneyim ve gerçekleştirilen hesaplamalar, çalışmanın devamında daha fazla metriğin de hesaplamaya katılabileceğini göstermesi bakımından önemli ve cesaret verici olmuştur. Bu prototipin ikinci özgün yönü ise Grasshopper içerisinde özel kullanıcı arayüzleri oluşturulabilmesine olanak sağlayan *Human UI* eklentisini kullanarak grafik arayüz özellikleri sunmasıdır. Şekil 9'da bu arayüzün bir ekranı gösterilmektedir. Bu prototipe ait diğer ekranların detaylı görüntüleri EK 4'te verilmiştir. Bu arayüz yardımıyla kullanıcılar Grasshopper kullanmayı bilmeseler bile yazılım eklentisinin bütün özelliklerini kullanabilmektedirler. Bu arayüz iki ana başlık içermektedir. Space Syntax isimli başlık, ağ yapısının modellenmesi ve analiz edilmesi ile ilgili işlevleri içermektedir. Setup ekranında düğüm noktaları isimlendirilmekte, Analyze ekranında ise çeşitli Mekan Dizimi analizleri gerçekleştirilmektedir. *Dynamic Graph* başlığının altında ise, Kangaroo ile gerçekleştirilen optimizasyonun kontrol edildiği ve ağ yapısındaki itme-çekme kuvvetlerinin hassas biçimde kontrol edilebildiği arayüz özellikleri bulunmaktadır. Bu prototip, bir Grasshopper kodu olmaktan öteye giderek ayrı bir yazılım eklentisi gibi çalışmaya başlayan ilk prototiptir.



Şekil 9. "SpaceChase for Grasshopper WIP" Prototip #4. Genel ekran görüntüsü. Bu prototipin diğer ekranları EK 4'te sunulmuştur.

Prototip #4, 3 ve 4 Haziran 2020 tarihlerinde dijital ortamda gerçekleştirilen bir atölyede test edilmiştir. Bu atölye çalışmasına ilişkin detaylı anlatıma Bölüm 4.3'te yer verilmiştir. Bu prototip Grasshopper'ın sunduğu hazır bileşenler ve eklentiler kullanılarak yapılabileceklerin sınırı olarak kabul edilmiştir. Tablo 2'de dört prototipin özellikleri yukarıda açıklanan alt iş paketleri üzerinden özetlenmiştir;

Tablo 2. "SpaceChase for Grasshopper WIP" prototiplerinin özelliklerinin karşılaştırılması.

Prototip	Ağ altyapısı	Dinamik ve etkileşimli simülasyon	Mekan Dizimi metrikleri
#1	Ağ verisi Rhinoceros ekranındaki çizgilerden alınmaktadır.	Hareket devamlı değildir. Kangaroo kullanılmıştır.	yok
#2	Ağ verisi csv formatındaki tablolardan alınmaktadır. Düğüm noktaları rastgele konumlanmaktadır.	Hareket devamlı değildir. Kangaroo kullanılmıştır.	Düğümler arasında en kısa yol hesabı yapabilir. Mekan Dizimi metriklerini hesaplayamaz.
#3	Ağ verisi csv formatındaki tablolardan alınmaktadır. Bağlamsal düğümler içerir.	Hareket devamlıdır. Kangaroo kullanılmıştır.	Bağlılık *(Connectivity)* değerini hesaplayabilir.
#4	Ağ verisi Rhinoceros ekranındaki çizgilerden alınmaktadır. Bağlamsal düğümler içerir.	Hareket devamlıdır. Kangaroo ve Human UI kullanılmıştır.	Bağlılık, Ortalama derinlik ve Entegrasyon değerlerini hesaplayabilir.

Yukarıda açıklanan prototipler, Grasshopper ve bağlantılı eklentilerin kullanıldığı denemelerdir. SpaceChase for Grasshopper'ın geliştirme stratejisinin ikinci aşaması; yukarıda özetlenen birinci adımda elde edilen bilgi ve deneyimin kullanıldığı yeni ve özgün bileşenlerden oluşan bir eklenti paketinin geliştirilmesidir. Birinci aşamada üretilen dört prototipten öğrenilenler aşağıdaki gibi özetlenebilir;

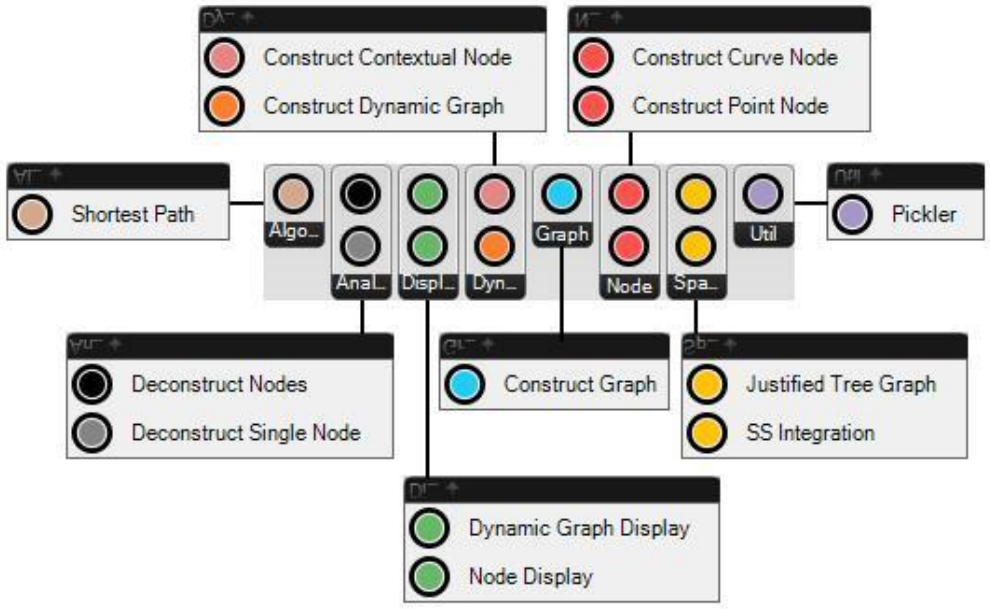
- Ağ altyapıları bağlamında değerlendirildiğinde, farklı ağ girdilerine olanak sağlayan bir esnekliğin gerekli olduğu görülmüştür. Eklentide hem Rhinoceros nesnelerinden hem de .csv gibi tablo verilerinden ağ türetmesinin yapılabilmesine karar verilmiştir. Dolayısıyla, geliştirilecek eklenti düğüm ve bağlantı verisini birden fazla biçimde tanımlayarak bunları aynı ağ yapısı içerisinde değerlendirebilecek bir teknik alt yapıyı kurmalıdır.
- Yapılan ilk çalışmalar, dinamik ve etkileşimli ağ yapısının Kangaroo tarafından sağlanabildiğini göstermiştir. Araştırmanın süresi ve teknik olanaklar değerlendirildiğinde, geliştirilecek eklentide Kangaroo'nun yerini alabilecek özgün bir optimizasyon algoritmasının uyarlanabilmesinin mümkün olmadığı görülmüştür. Çalışmanın Kangaroo dışında başka yöntemlerle dinamik ağ yapıları oluşturması potansiyeli ileri araştırmalara bırakılmıştır. Geliştirilecek eklentinin dinamik ağ yapısı bağlamında Kangaroo hedeflerinin türetilmesi üzerine kurulmasına karar verilmiştir.
- Mekan Dizimi metrikleri bağlamında, eklenti aşamasında prototip çalışmalarının geldiği seviyenin üzerine çıkılabileceği görülmüştür. Eklenti paketi geleneksel yazı-tabanlı bir programlama dili ile geliştirileceğinden, Grasshopper'ın yol açabildiği performans ve hesaplama sorunlarına çözümler üretilebileceği anlaşılmıştır. Böylelikle, Grasshopper içerisinde bu hesaplamaların daha hızlı ve kolay biçimde yapılmasına olanak sağlayacak yeni bileşenlerin geliştirilebileceği görülmüştür.
- Prototip #4, bir Grasshopper eklentisi olmanın ötesinde, kendi arayüzü olan ayrı bir yazılım denemesi gibi görülebilir. Fakat bu yaklaşım, Grasshopper'ı SpaceChase ile beraber farklı şekillerde kullanmayı kısıtlamıştır. Çalışmanın devamında bu yönde derinleşmek yerine, Grasshopper içerisinde çalışarak onun güçlü yönlerini kullanmaya devam edecek şekilde ilerlemeye karar verilmiştir. Dolayısıyla, ikincil arayüzler oluşturmak yerine, Grasshopper içerisine yerleştirilerek kullanılan bileşenler üretilecektir.

İkinci aşamanın ön çalışması olarak, hangi ortamların ve hangi kodlama dillerinin kullanılabileceği araştırılmıştır. Ön çalışmalar sonucunda iki temel dilin (ve kodlama ortamının) kullanılabileceği anlaşılmıştır. Bunlar; *C#* ve *Python* programlama dilleridir. *Microsoft Visual Studio* üzerinde Rhinoceros'un ilgili kütüphaneleri yüklenerek C# dilinde Grasshopper bileşenleri geliştirilebilmektedir. Bu yöntemin en önemli avantajları; C# dilinin getirdiği esnekliktir

ve Visual Studio'nun getirdiği hata ayıklama (*debugging*) olanağıdır. Visual Studio ortamında hazır ağ kütüphaneleri de dahil olmak üzere, pek çok farklı servis kullanılabilmektedir. Bu yöntemin gerektirdiği öğrenme süreci ve karmaşıklık düzeyi göz önüne alındığında, daha uzun süreli ve daha kapsamlı projelerde kullanılabilir olduğu anlaşılmıştır.

SpaceChase for Grasshopper'da kullanılan eklenti geliştirme yöntemi Python programlama diline dayanmaktadır. Bu yöntem, Grasshopper üzerinde hazır olarak bulunan Python Script bileşeni içerisine IronPython dilinde kodların yazılması ile hata ayıklama olanağı sunmakta ve sonrasında bu kodların Python dosyaları içerisinde birleştirilerek ve IronPython'un CLR modülü kullanılarak özgün Grasshopper bileşenlerini bulunduran .ghpy formatında bir eklenti dosyası halinde derlenmesine (*compile*) dayanmaktadır. Bu yöntem kullanılarak birinci yönteme kıyasla daha hızlı sonuç alınabilmektedir. Birinci yöntem kadar esnek ve açık olmayan bu yöntem, mevcut Rhino sürümünde sadece Windows bilgisayarlarda kullanılabilmektedir. Python programlama dilinin getirdiği kolaylıklar ve projenin süre sınırı göz önüne alınarak bu yöntemin kullanılmasına karar verilmiştir.

SpaceChase for Grasshopper'ın eklenti geliştirme sürecinde pek çok farklı sürüm geliştirilmiş ve test edilmiştir. Bu sürümler EK 11'de sunulan değişim günlüğünde (*changelog*) sunulmuştur. Bu bölümde bu sürümler içerisinde, çalışmanın kilometre taşları gibi görülebilecek üç tanesi açıklanacaktır. Bunlar 0.1.0, 0.2.5, ve 0.3.3 olarak numaralandırılmış Beta sürümleridir.

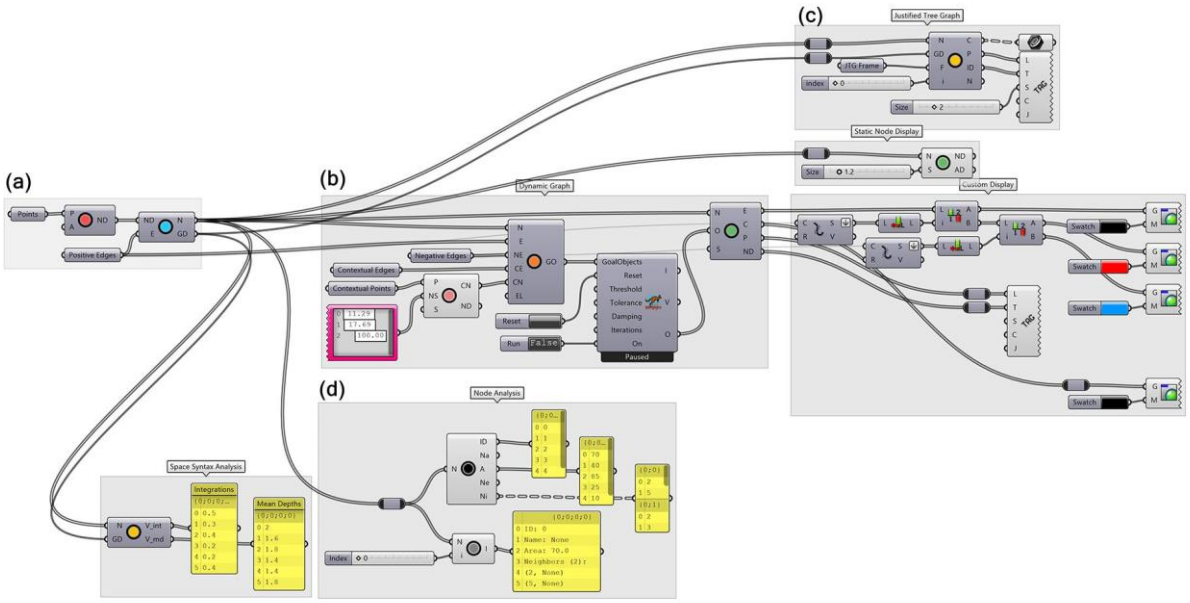


Şekil 10. "SpaceChase for Grasshopper Beta" (0.1.0) (Ağustos 2020). Eklenti sekmesi ve bileşen grupları. Bu bileşenlerin ayrıntılı açıklamaları EK 5'te sunulmuştur.

Şekil 10'de sunulan bileşenler eklentinin ilk ürünleridir. Bu raporda açıklanan prototip kodlarından öğrenilen teknik detaylar, iki aylık bir çalışmanın ardından eklentinin çalışan ilk sürümünün ortaya çıkmasını sağlamıştır. Çalışmanın öncesinde kullanılan programlama dili, Grasshopper bileşenlerinin nasıl oluşturulduğu ve kullanıldığına ilişkin araştırma ve denemeler gerçekleştirilmiştir. Eklentinin ilk sürümü sekiz bileşen grubundan ve bu grupların içerisindeki 13 bileşenden oluşmaktadır (Şekil 10 ve EK 5).

Eklentinin çalışma prensibi prototipler ile aynıdır. Prototiplerden farkı, Grasshopper'ın içerdiği hazır bileşenler yerine, proje için geliştirilmiş özel amaçlı bileşenler ve veri türleri ile çalışıyor olmasıdır. SpaceChase'in oluşturduğu ve kullandığı veri türleri, Grasshopper diline uygun biçimde çeşitli harf sembolleri ile ifade edilen program nesneleri (*objects*) tanımlamıştır. Bu program nesneleri, geliştirilen yeni bileşenlerin girdi (*input*) ve çıktı (*output*) parametrelerinde görülebilmektedir. Eklentinin ilk sürümü olan 0.1.0'in sürece en önemli katkısı, bu veri türlerinin tanımlanması ve ağ yapısına ilişkin matematiksel bilgileri veri akış diyagramı boyunca program nesneleri vasıtasıyla taşıyabilir hale getirmesidir. SpaceChase'in temel program nesneleri aşağıdaki gibidir;

- *Node Data* (ND): Bu program nesnesi, düğüm noktaları ile ilgili verileri taşımaktadır. Prototiplerden elde edilen deneyim, farklı yöntemlerle düğüm noktaları oluşturulabileceğini göstermiştir. Her yöntemin avantajları ve dezavantajları olduğu için, SpaceChase eklentisinde *Node Data* oluşturabilen birden fazla yöntem, ayrı ayrı bileşenler tanımlamıştır.
- *Edge* (E): SpaceChase eklentisinde bağlantılar *Edge* girdi ve çıktı parametreleri olarak isimlendirilmektedir. Taşıdıkları bilgi, Rhinoceros'un çizgi nesneleridir. Ağ üzerinde negatif ve pozitif etkilerin (itme-çekme) niteliği başka bileşenler tarafından oluşturulmakta, bu bilgi *Edge* nesnelerinde tutulmamaktadır. Ağ üzerindeki etkilerin (itme-çekme) niceliği ise *Edge* nesnelerinin içerdiği çizgilerin uzunluk bilgisi vasıtasıyla aktarılmaktadır.
- *Nodes* (N): Bu program nesnesi, *Node Data* ve *Edge* nesnelerinin birleşimiyle oluşturulur ve SpaceChase'in ürettiği ağ yapılarındaki düğüm noktalarını temsil eder. Mekan Dizimi metriklerinin hesaplanmasında ve dinamik ağ yapılarının oluşturulmasında kullanılır. Dolayısıyla SpaceChase'in çalışma prensibinde ilk aşama *Node* nesnelerinin tanımlanmasıdır.
- *Graph Data* (GD): SpaceChase eklentisinin ürettiği ağ yapısıdır. Bu program nesnesi üretildikten sonra hem Mekan Dizimi metriklerinde hem de dinamik ağ yapısının oluşturulmasında kullanılmaktadır. Ağ topolojisi ile ilgili gereken bilgilerin *Node Data* ve *Edge* nesnelerinden alınarak düzenlendiği, paketlendiği bir nesnedir.

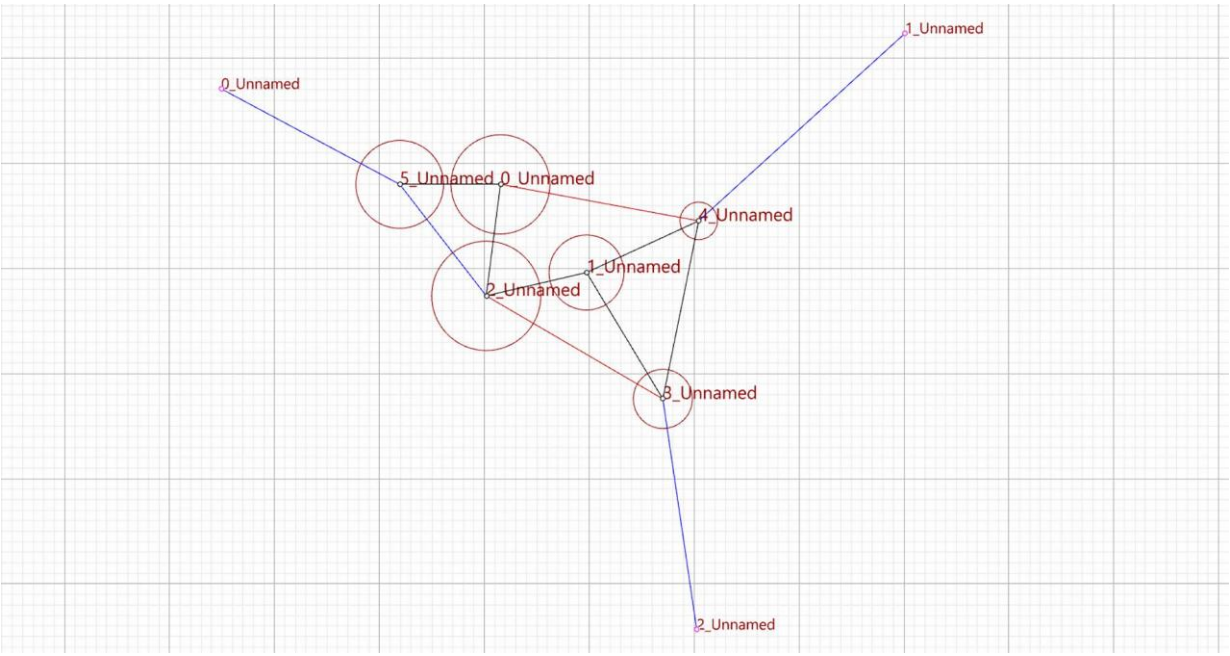


Şekil 11. "SpaceChase for Grasshopper Beta" (0.1.0); örnek veri akış diyagramı. Bu kod Ek 6'da ayrıntılı olarak verilmiştir.

Şekil 11'de eklentinin ilk sürümü ile yapılmış bir örnek Grasshopper kodu görülmektedir. Prototip aşamasında düğüm noktalarının oluşturulması için birden fazla yöntemin kullanılabileceği görülmüştür. Örneğin .csv tablosundan verileri okumak veya Rhinoceros'taki geometrik nesnelerden bilgi almak mümkündür. Bu yöntemler eklentide ayrı bileşenler halinde kodlanmıştır. Bu amaçla geliştirilen her bileşenin çıktısı (*Node Data* ve *Edge*) aynı olmakla beraber, girdi parametreleri farklıdır. Şekil 11'de düğüm noktalarının Rhinoceros çizgi ve noktalarından alındığı standart iş akışı görülmektedir. SpaceChase'in ilk sürümünden itibaren geliştirilen ve son sürümde de kullanılan ana bileşenlerden birisi *Construct Graph* bileşenidir. Şekil 11-a'da mavi renkte görülen bu bileşen, ağ yapısı ile ilgili bütün bilgileri (düğüm noktalarının pozisyonları, isimleri, bağlantıları vb.) toplayarak SpaceChase'in kullanabileceği ağ yapısına (*Graph Data*) dönüştürmekten sorumludur.

SpaceChase ağ yapısı oluşturulduktan sonra bu yapı iki farklı şekilde kullanılabilmektedir. Bunlardan ilki dinamik olmayan, sabit ağ yapısı üzerinde Mekan Dizimi analizlerinin ve görselleştirmelerinin yapılmasıdır (Şekil 11-d). Diğeri ise, dinamik ağ yapısının oluşturulması ve Kangaroo vasıtasıyla simüle edilmesi ve görselleştirilmesidir (Şekil 11-b ve c). Şekil 11'de görüldüğü gibi, bu iki kullanım şekli (bileşen grubu) *Construct Graph*'tan aldıkları ağ verisi ile birbirlerinden bağımsız olarak çalışabilmektedirler. Bu durum proje boyunca tartışılan bazı avantajları ve dezavantajları ortaya çıkarmıştır. Eklentinin ilk sürümünden itibaren iki farklı kullanım şeklinin olabileceği anlaşılmış ve bu bir zenginlik ve esneklik olarak görülmüştür. Bu kullanım şekilleri, eklentinin son sürümünde de bulunmaktadır ve Ek 10 ve Ek 12'de daha detaylı

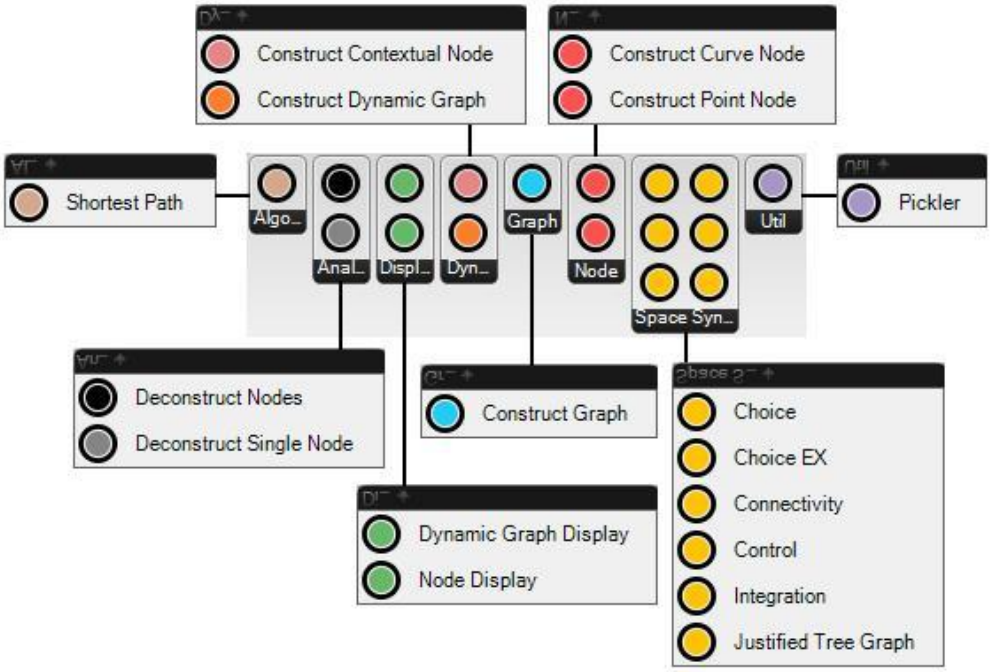
olarak açıklanmıştır. Diğer yandan, Mekan Dizimi metriklerinin analizi ile dinamik ağ simülasyonunun birbirinden ayrı çalışmaları projenin ana hedefleri bağlamında bazı dezavantajları da beraberinde getirmiştir. İlk sürümlerde Mekan Dizimi metriklerinin dinamik ağ ile ilişkisinin kopuk olması projenin alana özgün katkısı olan dinamik ağ yapıları ile ilgili sorunları gündeme getirmiştir. Eklentinin son sürümünde geliştirilen *Sentaktik Ağ Modeli*, bu soruya kısmen cevap vermiştir.



Şekil 12. “SpaceChase for Grasshopper Beta” (0.1.0). Örnek ekran görüntüsü.

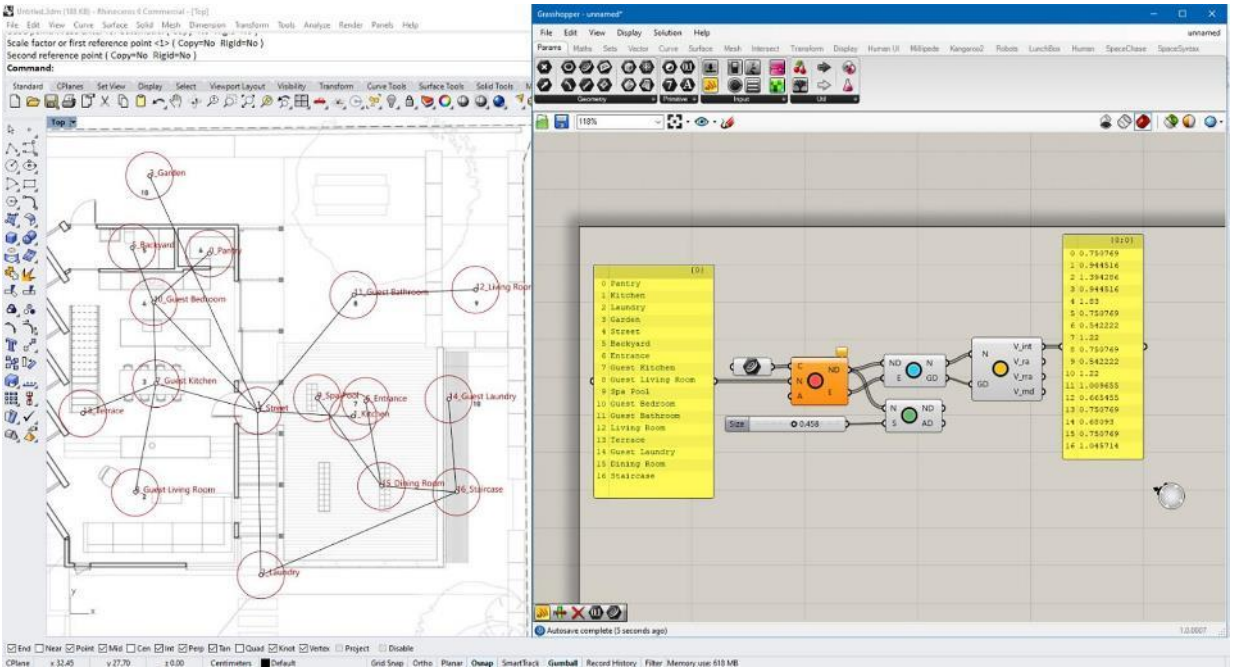
Şekil 12’de eklentinin ilk sürümü ile ilgili bir örnek ekran görüntüsü sunulmuştur. Eklentinin bundan sonraki gelişim sürecinde ekranda görülen sonuçlarda önemli bir farklılık gözlemlenmemiştir. Eklentinin işlevlerinin zenginleştirilmesi ve karşılaşılan program hatalarının giderilmesi üzerine yoğunlaşılmıştır. Sürdürülen geliştirme ve hata ayıklama (*debugging*) çalışmaları sonucunda Şekil 13’te görülen, sürecin ikinci kilometre taşı olan 0.2.5 numaralı sürüm ortaya çıkarılmıştır. Bu sürüm 8 bileşen grubu altında 17 bileşen bulundurmaktadır. Bu bileşenlerin işlevleri, girdi ve çıktı parametreleri Ek 7’de detaylı olarak sunulmuştur. Önceki sürümde sorunsuz çalışan bileşenler bu sürümde de kullanılmıştır. Eklentinin temel kullanım şeklinde bir değişiklik olmamıştır. Dolayısıyla sadece bir önceki sürümden farklı olarak gerçekleştirebildiği yeni işlevler özetlenecektir. Ek 11’de sunulan değişim günlüğü (*changelog*)’da görüldüğü gibi 0.2.x numaralı sürümlerin eklentiye kattığı en önemli yenilik, Mekan Dizimi analizleri ile ilgilidir. Bu aşamada SpaceChase’in ürettiği ağ yapılarını daha zengin biçimde analiz edebilmesine odaklanılmıştır. Bu araştırma Mekan Dizimi metrikleri ile ilgili bilinen matematiksel yöntemlerin Python diline aktarılması sonucunda özgün kodların geliştirilmesini

sağlamıştır. Geliştirilen hesaplama metrikleri Şekil 13'te sarı renkte görülmektedirler. Bu bileşenlerin açıklamaları Ek 7'de sunulmuştur. Metrikler geliştirildikten sonra yaptıkları hesaplamaların doğruluğu Syntactic yazılımı kullanılarak onaylanmıştır.

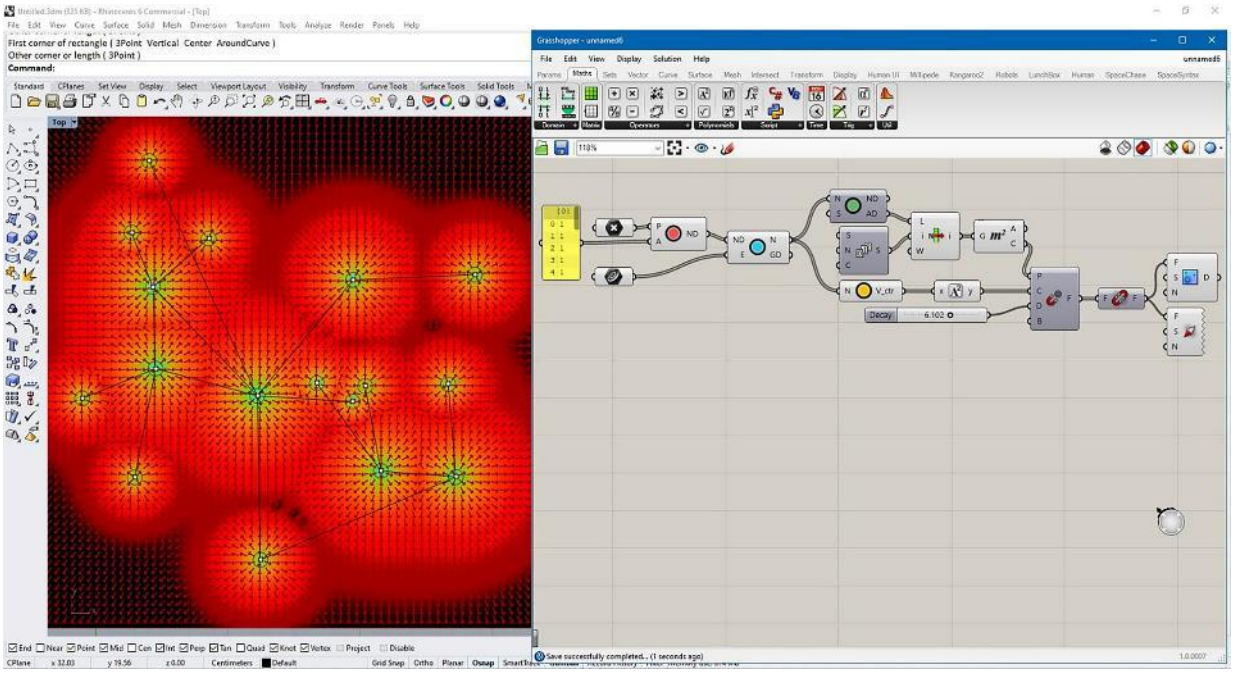


Şekil 13. "SpaceChase for Grasshopper Beta" (0.2.5) (Eylül 2020). Eklenti sekmesi ve bileşen grupları. Bu bileşenlerin ayrıntılı açıklamaları EK 7'de sunulmuştur.

Bu sürüm, *Education and Research in Computer-aided Architectural Design in Europe* (eCAADe) 2020 konferansı kapsamında uluslararası bir ortamda gerçekleştirilen *Atölye III* ile test edilmiştir (Bkz. Bölüm 4.3). Eklentinin bu sürümünün ekran görüntüsünde önemli bir değişiklik olmamıştır. Atölye sırasında eklentinin görselleştirme kapasitesi üzerine bazı denemeler yapılmıştır. Şekil 14, 15 ve Ek 8'de bu denemelerden bazıları görülmektedir. Şekil 14'te Atölye III esnasında yapılan bir egzersiz görülmektedir. Bu egzersizde bir plan çizimi üzerinden dinamik olmayan ağ yapısının oluşturulması ve entegrasyon değerlerinin hesaplanması test edilmiştir. Şekil 15'te görülen çalışmada ise SpaceChase'in Grasshopper'ın başka özellikleri ile beraber kullanılması denenmiştir. Grasshopper'ın vektör alanları üreten bileşenleri ağ yapısı ile bütünleştirildiğinde düğüm noktaları yerine alanlar üzerinden çalışan bir algoritma oluşturmak mümkündür.



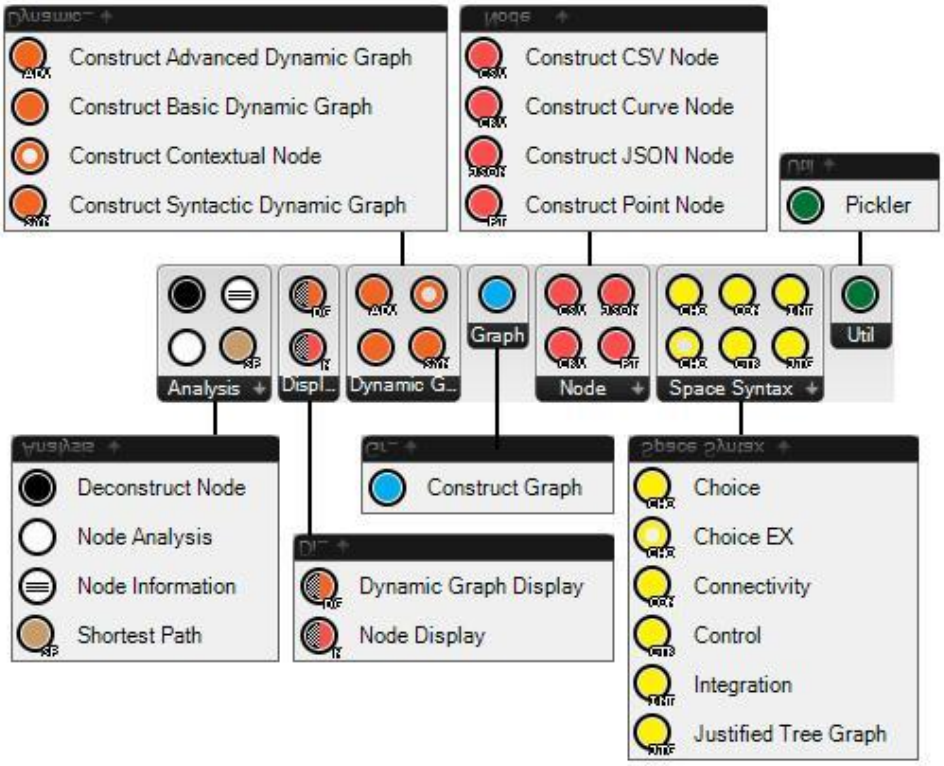
Şekil 14. “SpaceChase for Grasshopper Beta” (0,2,5); örnek veri akış diyagramı ve ekran görüntüsü.



Şekil 15. “SpaceChase for Grasshopper Beta” (0.2.5). Ekran görüntüsü. Eklentinin görselleştirme kapasitesinin test edildiği farklı denemeler Ek 8’de sunulmuştur.

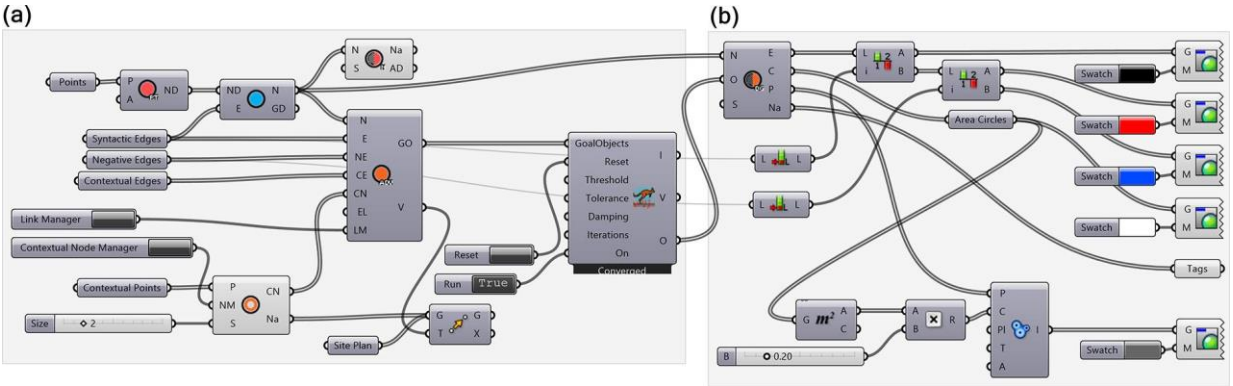
SpaceChase’in araştırma projesi sürecinde geliştirilen son sürümü 0.3.3’tür. Şekil 16’da görüldüğü gibi, eklentinin bu sürümü 7 bileşen grubu içerisinde toplam 22 bileşen barındırmaktadır. Bu sürümde mevcut hesaplamalar ve bileşenlerde tespit edilebilmiş bir hata

kalmamış olup, sürüm eklentinin dağıtıma hazır (*release*) versiyonu (1.0.0) sayılabilecek niteliktedir.



Şekil 16. "SpaceChase for Grasshopper Beta" (0.3.3) (Kasım 2020). Eklenti sekmesi ve bileşen grupları. Bu bileşenlerin ayrıntılı açıklamaları EK 9'da sunulmuştur.

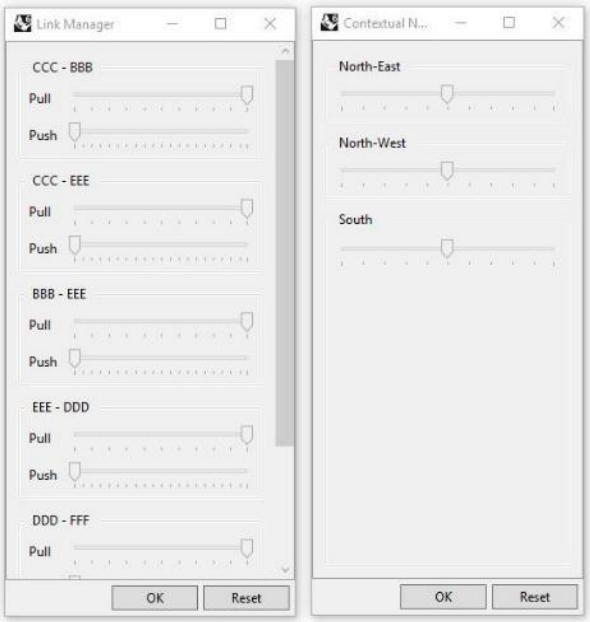
Eklentinin kullanım biçimi detaylı olarak bölüm 4.1.3'te açıklanmıştır. Grasshopper'ın kullanım biçimini de kapsayan daha detaylı bir kullanım kılavuzu ise Ek 12'de sunulmuştur. Bu bölümde bu sürümün bir öncekinden (0.2.5) farkları ifade edilecektir.



Şekil 17. "SpaceChase for Grasshopper Beta" (0.3.3); örnek veri akış diyagramı. Bu kod Ek 10'da daha ayrıntılı olarak verilmiştir.

Eklentinin son sürümünde, çok sayıda hata ayıklama ve güncellemenin dışında, farklı kaynaklarla veri alışverişinin zenginleştirilmesine ve atölyelerden gelen geri bildirimler sonucunda bazı arayüz özelliklerinin eklenmesine karar verilmiştir. İlk prototiplerinden itibaren zaman zaman kullanılan, diğer ağ görselleştirmesi yazılımlarında yaygın olarak kullanılan .csv formatındaki tablolardan veri alma özelliği eklenmiştir. Böylelikle eklentinin ağ yapıları üzerine çalışan benzer yazılımlarla (Cytoscape, Syntactic, Gephi vb.) veri alışverişi gerçekleştirmesi kolaylaşmış ve bu yazılımların kullanıcılarının SpaceChase'i daha kolay anlaması ve kullanabilmesi sağlanmıştır. Benzer şekilde, SpaceChase oyun yazılımının ürettiği .json formatındaki veriyi okuyan bir bileşen geliştirilmiş, böylece oyun ortamında elde edilen ağ yapılarının SpaceChase, dolayısıyla Rhinoceros ve Grasshopper ortamlarına taşınması mümkün hale gelmiştir. Sadece bu özelliğin bile, dinamik ağ özelliklerinin kullanılmadığı durumda, SpaceChase oyunu ve Rhinoceros ortamı arasında kurduğu bağlantı nedeniyle kayda değer bir adım olduğu değerlendirilmiştir.

Eklentinin son sürümünde dinamik ağ modeli yaklaşım seviyesi bakımından üç ayrı yönteme (bileşene) ayrılmıştır. Bunlar yazılım eklentisinin temel kullanım biçimini belirlemektedir. *Basic, Advanced* ve *Syntactic* olarak ayrılan bu üç ağ modeli, bölüm 4.1.3 ve Ek 12'de detaylı olarak incelenmiştir. Özetle bu üç model tipi, düğüm noktaları ve farklı nitelikteki bağlantı verisini farklı şekillerde girdilerle destekleyerek Kangaroo hedeflerine dönüştürmektedirler. *Construct Syntactic Dynamic Graph*, daha önce ifade edilen, dinamik ağa Mekan Dizimi metriklerinin katılımı ile ilgili konuda önemli bir gelişmedir. Bu yöntem (ve bileşen) kullanılarak Kangaroo'ya aktarılacak hedefler içerisine kullanıcının belirlediği Mekan Dizimi metrikleri de katılabilmektedir. Fakat bu yöntemin bir atölye çalışması ile test edilmesi proje süresi içinde mümkün olamamıştır.



Şekil 18. "SpaceChase for Grasshopper Beta" *Advanced Graph* modelinin sunduğu "*Contextual Node Manager*" ve "*Link Manager*" arayüzleri.

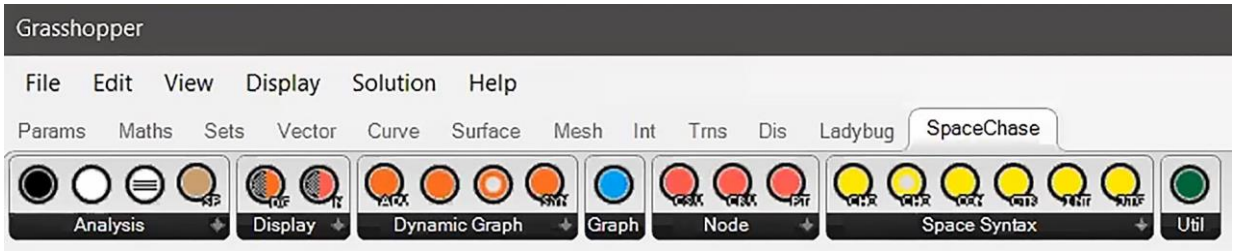
Yazılım eklentisinin son sürümünde gerçekleştirilen bir diğer yenilik de bağlantıların hassas biçimde kontrol edilebilmesi ile ilgilidir. Yapılan denemelerde kullanıcıların dinamik ağ yapısının hareket biçimi üzerinde yeterince kontrol sahibi olamadıkları görülmüştür. Bağlamsal düğümlerin ağ üzerindeki etkisine ve düğümler arası bağlantı uzunluklarına Grasshopper üzerinden erişim sağlamak mümkündür; fakat bu çalışma düzeni yeterince esnek değildir. Kullanıcı etkileşimini güçlendirmek için eklentiye Şekil 18'de görülen iki pencere eklenmiştir. Bunlardan ilki dinamik ağ yapısında bulunan bağlantıları listelemekte ve her bağlantının gücünün kaydırma çubukları kullanılarak kontrol edilebilmesine olanak sağlamaktadır. İkinci pencere ise, benzer şekilde bağlamsal düğüm noktalarının çekme etkisini daha hızlı biçimde kontrol etmeyi mümkün kılmaktadır. Bu pencerelerin kullanım şekli bölüm 4.1.3 ve Ek 12'de daha detaylı olarak incelenmiştir.

Mevcut yazılım eklentisi, Mekan Dizimi teorisinin mimarlar ve öğrenciler tarafından kullanılmasına, dinamik ve etkileşimli bir ortamda eskiz anlayışıyla ilişki şemalarının oluşturulmasına, bu şemalara müdahale edilip izlenebilmesine ve gerçek-zamanlı olarak analizlerin yapılabilmesine olanak sağlamaktadır. Ancak bu dinamik ve etkileşimli ortam, potansiyelinin aksine, henüz mevcut Mekan Dizimi metriklerinin dışında dinamik ağ yapısına özgün bir hesaplama yapamamaktadır. SpaceChase for Grasshopper'ın sınırları ve potansiyelleri daha detaylı biçimde 4.1.4 numaralı bölümde ele alınacaktır.

4.1.3 Temel Kullanım Biçimi

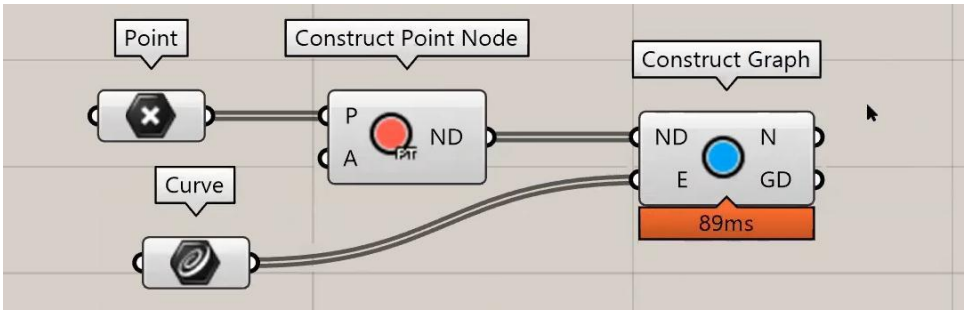
Bu bölümde SpaceChase for Grasshopper eklentisinin son sürümünün (0.3.3) nasıl kullanıldığı ve temel özellikleri özetlenmiştir. Bu sürümün içerdiği bileşenlerin girdi ve çıktı parametreleri ile kullanım amaçlarını içeren bileşen işlev tablosu Ek 9'da sunulmuştur. Bu bölümde anlatılan işlemlerin daha detaylı ve teknik açıklamaları Ek 12'deki *Kullanım Kılavuzu*'nda görülebilir.

SpaceChase for Grasshopper'ın çalıştırılabilmesi için Windows tabanlı bir bilgisayarda Rhinoceros 6 (SR25 veya sonrası) yüklü olmalıdır. Eklenti Rhinoceros 7 altında da çalışabilmektedir. Eklentinin son sürümü olan 0.3.3 (10.01.2021 itibariyle), *http://spacechase.app* web adresinden indirilebilir. Yükleme sorunsuz tamamlandığında Grasshopper penceresinde en üst satırdaki komut sekmelerinde *SpaceChase* isimli bir sekme belirmelidir (Şekil 19).



Şekil 19. SpaceChase sekmesinde bulunan bileşenler 7 grup içerisinde organize edilmiştir.

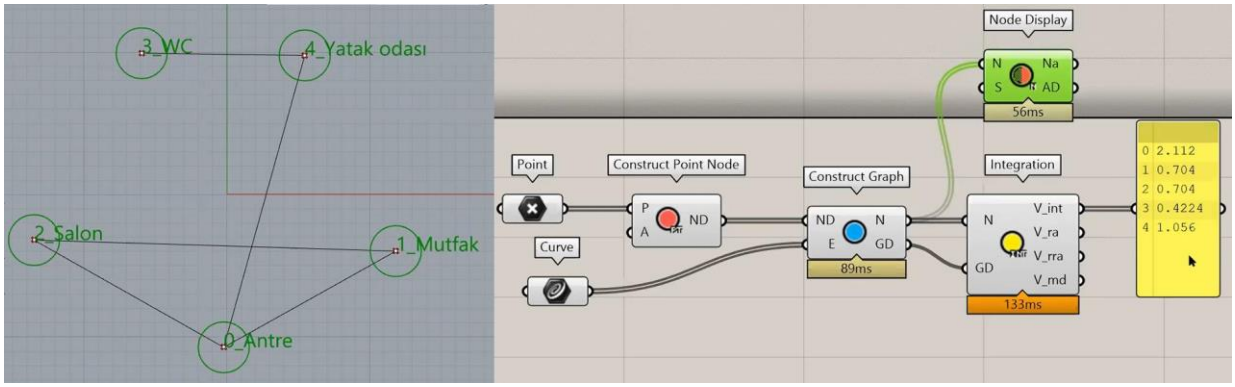
SpaceChase for Grasshopper ile çalışmaya başlamadan önce ağ yapısını tanımlamak gerekmektedir. Rhinoceros üzerinde düğüm noktalarının ve bağlantıların tanımlanarak Grasshopper içerisine aktarılmasının ardından SpaceChase'in kullanımına geçilebilir. SpaceChase'in düğüm noktaları *Node* (N) olarak isimlendirilmiştir. Düğüm noktalarını oluşturan verilerin (koordinat, isim, ağırlık vb.) toplamı da *Node Data* (ND) veri tipi olarak tanımlanmıştır. Nokta verisini SpaceChase'in okuyup işleyebileceği düğüm noktalarının verisine (ND) dönüştürebilmek için SpaceChase sekmesindeki *Node* başlıklı bileşen grubu altındaki bileşenler kullanılmalıdır. Bu bileşenlerin hepsi farklı girdileri kullanarak *Node Data* (ND) verisi türetmektedir. SpaceChase'in bir diğer ana bileşeni, *Node Data* ve *Edge* (E, bağlantı) verisini alarak SpaceChase ağ yapısına (GD) dönüştüren *Construct Graph* bileşenidir. Bu bileşenler kullanılarak basit bir ağ yapısı Grasshopper içerisinde modellenebilmektedir (Şekil 20).



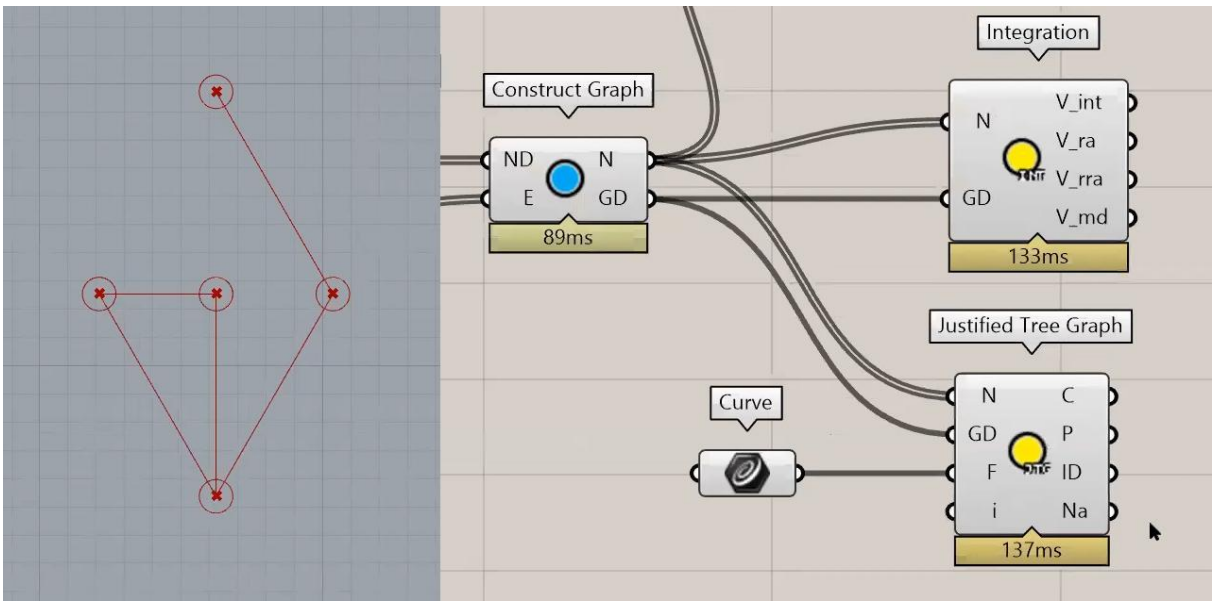
Şekil 20. *Construct Graph* bileşeni ile düğüm (ND) ve bağlantı (E) verilerinden SpaceChase düğüm (N) ve ağ (GD) verilerinin türetilmesi.

Bu aşamadan sonra SpaceChase for Grasshopper'ın iki temel kullanım şekli bulunmaktadır. Eğer hareketli ve etkileşimli bir simülasyona gerek duyulmuyorsa, yukarıda açıklanan bileşenler yardımıyla oluşturulan ağ yapıları doğrudan Mekan Dizimi analizlerine sokulabilmektedir. SpaceChase sekmesindeki *Space Syntax* başlıklı bileşen grubunda bulunan bileşenler GD veri türündeki ağların seçim (*Choice*), bağlılık (*Connectivity*), derinlik (*Depth*), kontrol (*Control*), entegrasyon (*Integration*) değerlerini hesaplayabilmektedir (Şekil 21). Ayrıca ağ yapısını görselleştiren *Justified Graph* oluşturulabilmektedir (Şekil 22). SpaceChase for Grasshopper'ın ayırt edici özelliklerinden birisi de analiz sonuçlarının ve ağ görselleştirmesinin tümüyle

Grasshopper ile uyumlu çıktılar sunmalarıdır. Böylece tasarım ve görselleştirme bağlamında Grasshopper'ın olanakları kullanılmaya devam edilebilmektedir. Bu potansiyelin erken bazı örnekleri (bir önceki sürümden) EK 8'de sunulmuştur.



Şekil 21. Düğüm noktalarının görselleştirilmesi ve entegrasyon değerlerinin okunması.

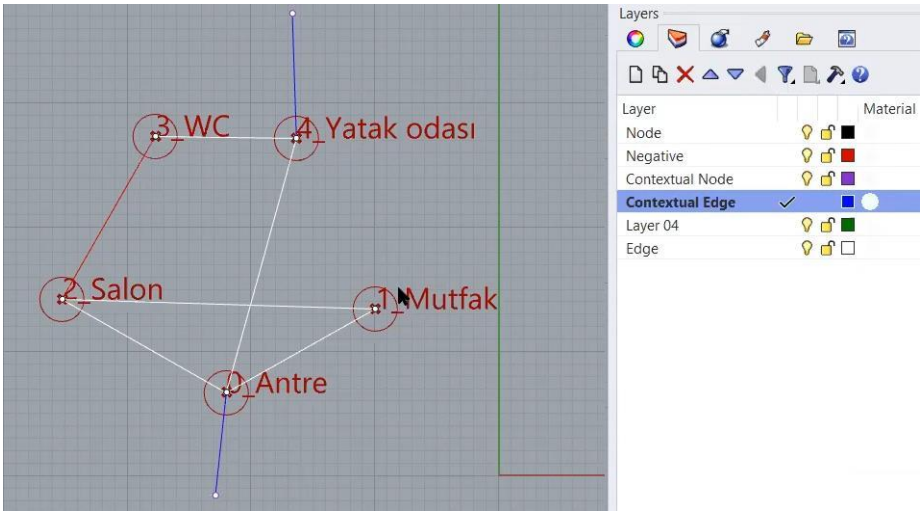


Şekil 22. *Justified Tree Graph* oluşturulması.

Eklentinin ikinci temel kullanım şekli, dinamik ve etkileşimli ağ yapılarıdır. Dinamik ağ yapısında düğüm noktalarının bir simülasyon içerisinde devinime girmesi beklenmektedir. Simülasyonu kontrol edebilmek için sabit konumda kalması istenen, sentaktik hesaplamalara katılmayan, diğer düğüm noktalarının konumunu ve hareket biçimini etkilemesi istenen özel düğüm noktaları tanımlanmıştır. Bu noktalar bağlamsal düğüm *(Contextual Node)* olarak isimlendirilmiştir.

Dinamik ağ yapılarında bağlantılar, aynı zamanda ağın hareketini de belirleyen unsurlardır. Dolayısıyla, SpaceChase içerisinde negatif ve pozitif bağlantılar tanımlanabilmektedir. Negatif değerdeki bağlantılar, iki ucundaki düğüm noktalarını birbirinden uzaklaştırma eğilimindedir.

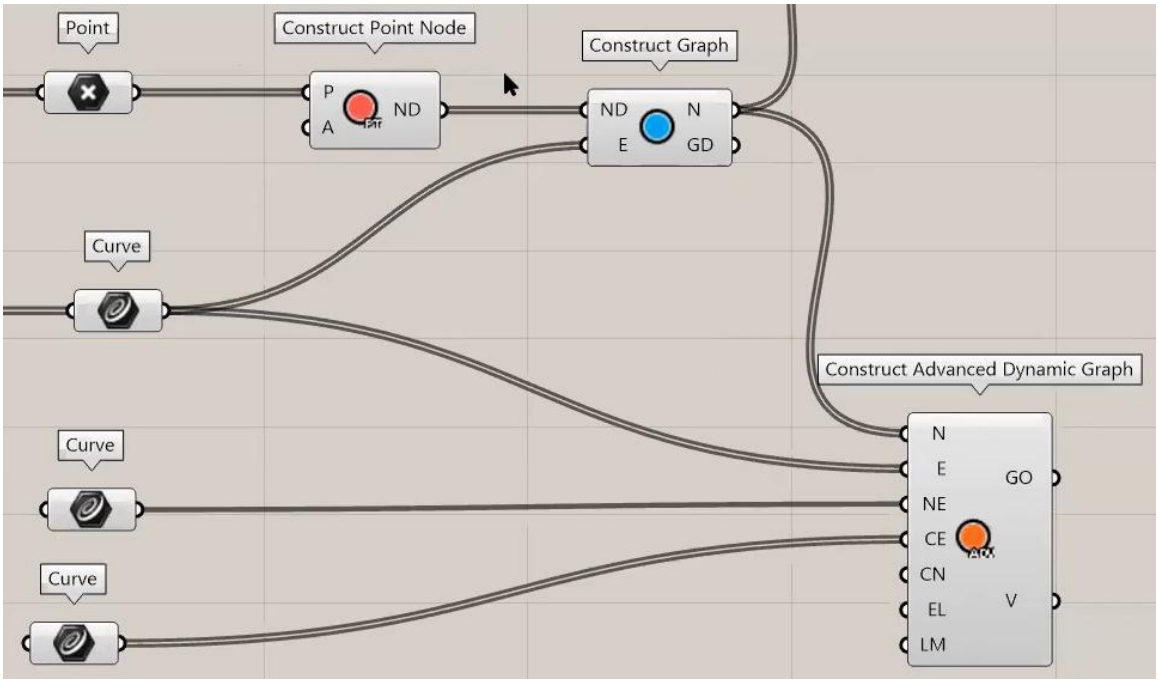
Pozitif değerdeki bağlantılar ise iki ucundaki düğüm noktalarını birbirine yaklaştırma eğilimindedir. Pozitif ve negatif bağlantılar, bağlamsal düğüm noktaları gibi sentaktik analizlere girmezler, dolayısıyla sadece simülasyonun hareket biçimini belirlemek için kullanılırlar.



Şekil 23. Bağlamsal düğüm noktalarının, negatif bağlantıların (kırmızı) ve bağlamsal bağlantıların (mavi) tanımlanması ve Rhinoceros'ta farklı katmanlarda çizilmesi.

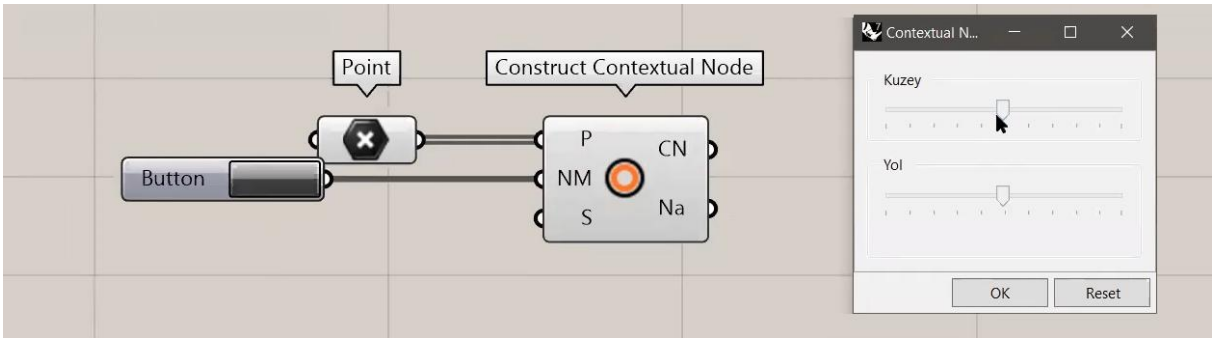
Ön hazırlıklar tamamlandıktan sonra ağ yapısının dinamik simülasyonuna geçilebilir. SpaceChase'in *Dynamic Graph* başlıklı bileşen grubu içerisinde bulunan bileşenler farklı amaçlar için çözümler sunmaktadır. Üç çeşit dinamik ağ modeli bulunmaktadır. Bunlar;

- **"*Basic*" ağ modeli:** Hızlı dinamik ağ görselleştirmesinde kullanılmaktadır. Grasshopper ortamında yapılamayan döngüsel hesaplamaları kendi içerisinde yapan bu bileşen çizgi ve nokta verilerini pozitif/negatif ilişkiler, uzunluk ve mesafeler bakımından hesaplayarak bir dinamik ağ modeli üretir.
- **"*Advanced*" ağ modeli:** İkinci ağ modeli *Basic* modeldeki hesaplamalara bireysel veya toplu olarak müdahale edilebilme imkanı sağlar. Benzer hesaplamalar yapan ancak müdahale imkanını bir arayüz vasıtasıyla kullanıcı dostu bir şekilde sunabilmek için sıfırdan kodlanan ikinci ağ modeli (Şekil 24), hazırlık sürecinin daha fazla vakit gerektirmesi nedeniyle birinci modelden ayrı tutulmuştur.
- **"*Syntactic*" ağ modeli:** Bu ağ modeli ikinci modelde kullanıcının insiyatifinde olan hesaplamaları, bileşene girdi olarak seçilen metrikler yardımıyla yapar ve "*sentaktik*" bir dinamik ağ görselleştirmesi sunar. Ağırlık (*weight*) ve güç (*strength*) olarak tanımlanan dinamik ağ özelliklerini kullanıcının seçtiği metrikler karşılığında sistem içerisinde göreceli olarak hesaplayan bileşen, Mekan Dizimi metriklerinin görsel bir temsilini sunmayı amaçlar.



Şekil 24. *Construct Advanced Dynamic Graph* bileşeninin temel bağlantıları.

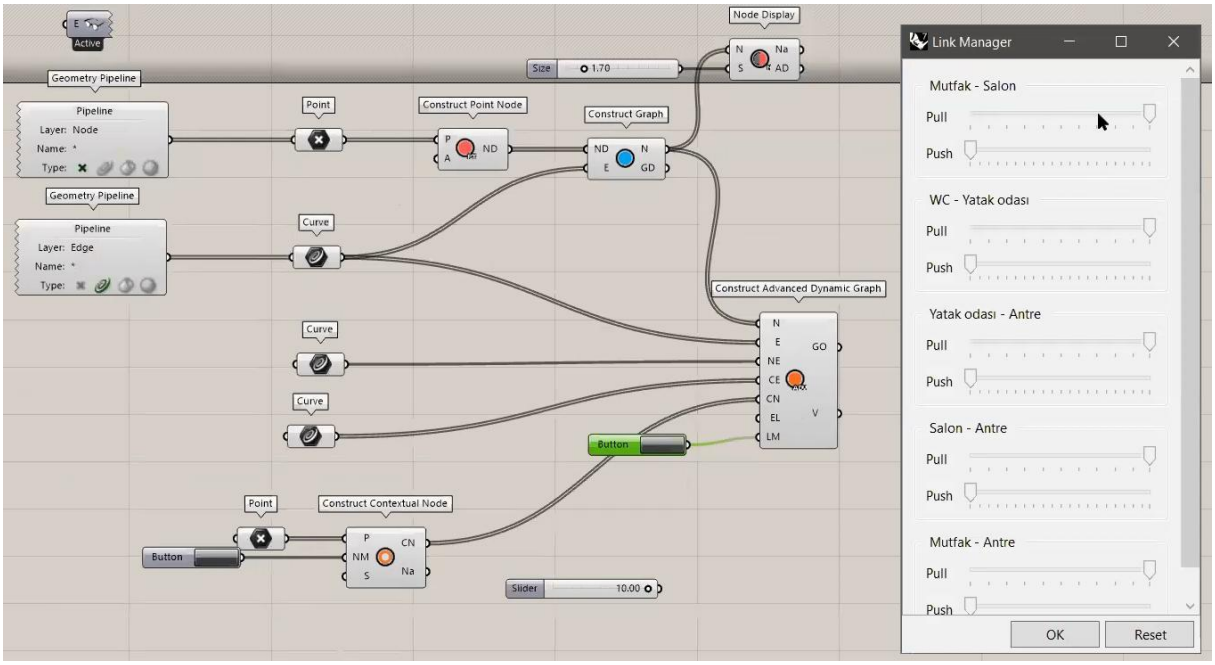
Construct Advanced Dynamic Graph bileşenine bağlamsal düğüm noktalarının (*Contextual Node*) aktarımı bir ara bileşen yardımıyla yapılmaktadır. Bu nesneleri SpaceChase'in işleyebileceği bağlamsal düğümlere dönüştürmek için *Construct Contextual Node* bileşeni kullanılır. Normal düğüm noktalarından farklı olarak bu bileşenin NM isimli bir girdisi bulunmaktadır. NM girdisi bir *Contextual Node Manager* isimli pencereyi açan bir düğme görevi görmektedir. *Contextual Node Manager*'de bulunan kaydırma çubukları ile bağlamsal düğüm noktalarının etki mesafelerini düzenlemek mümkündür (Şekil 25).



Şekil 25. *Construct Contextual Node* bileşeni ve *Contextual Node Manager* penceresi.

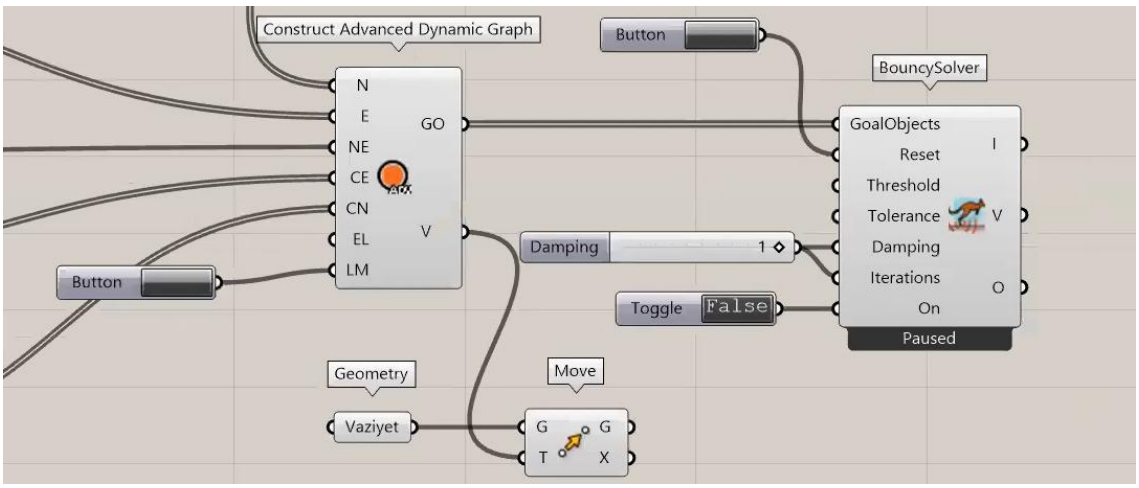
Construct Advanced Dynamic Graph bileşeninin son iki girdisinden biri olan *Edge Length* (EL) ağ yapısındaki bütün bağlantıların uzunluklarını belirli bir sayı değerine eşitlemek için kullanılan opsiyonel bir girdidir. Bu bileşenin LM girdisi bağlanan bir düğmeye basılıp aktifleştirildiğinde *Contextual Node Manager*'de olduğu gibi *Link Manager* isimli bir ayarlama penceresi açacaktır.

Bu pencere kullanılarak ağ yapısındaki bütün bağlantıların itme ve çekme etkileri hassas biçimde kontrol edilebilmektedir (Şekil 26).



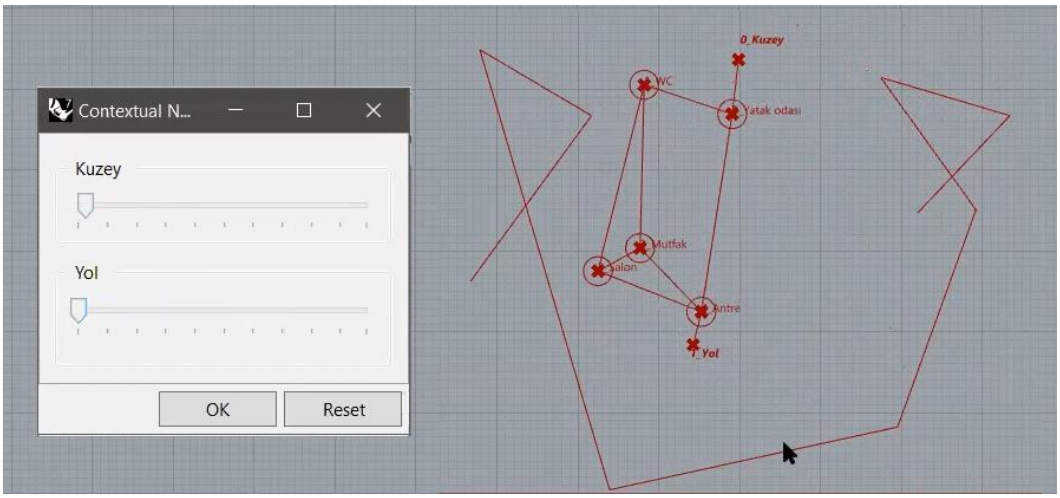
Şekil 26. *Construct Advanced Dynamic Graph* bileşeninin bağlantıları ve *Link Manager* penceresi.

Bu bağlantılar ile *Construct Advanced Dynamic Graph* bileşeni hareketli ağ için gereken bütün girdileri toplamış olacaktır. Bu bileşenin birinci çıktısı olan GO, Kangaroo hedefleridir. Kangaroo bileşenleri içerisindeki *Bouncy Solver* bileşeninin *Goal Objects* girdisine, *Construct Advanced Dynamic Graph* bileşenin GO çıktısı bağlanmalıdır. *Reset* girdisine bir *Button* bileşeni, *On* girdisine de Grasshopper'ın *Toggle* bileşeni bağlanmalıdır (Şekil 27).

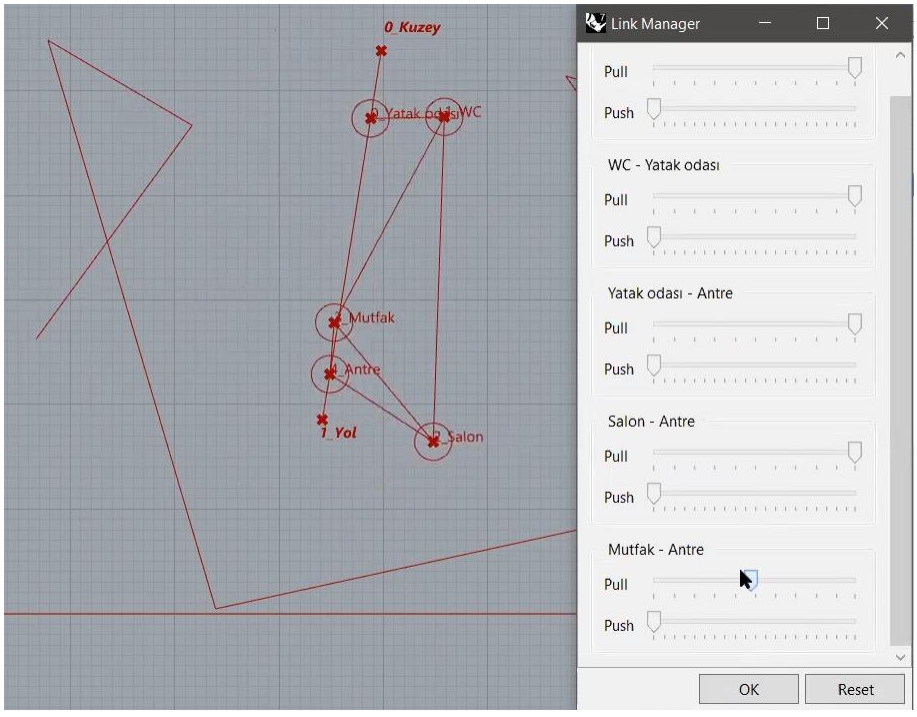


Şekil 27. Kangaroo *Bouncy Solver* bileşenine girdilerin aktarılması ve hareketli ağ için ayarlanması.

Bouncy Solver bileşeninin *On* girdisine bağlı *Toggle* üzerine çift tıklanarak *True* pozisyonuna getirildiğinde Rhinoceros ekranında hareketli ağ yapısı oluşmalıdır. Bu hareket esnasında görüntülenen bağlantılar tüm itme ve çekme ilişkilerini içermektedir. Bu temel kurulum üzerinde çalışmaya devam edilerek tasarım senaryosuna ve kullanım amacına uygun Grasshopper algoritmaları geliştirmek mümkündür. *Contextual Node Manager* penceresi kullanılarak bağlamsal düğüm noktalarının ağ üzerindeki etkileri kontrol edilebilmektedir (Şekil 28).

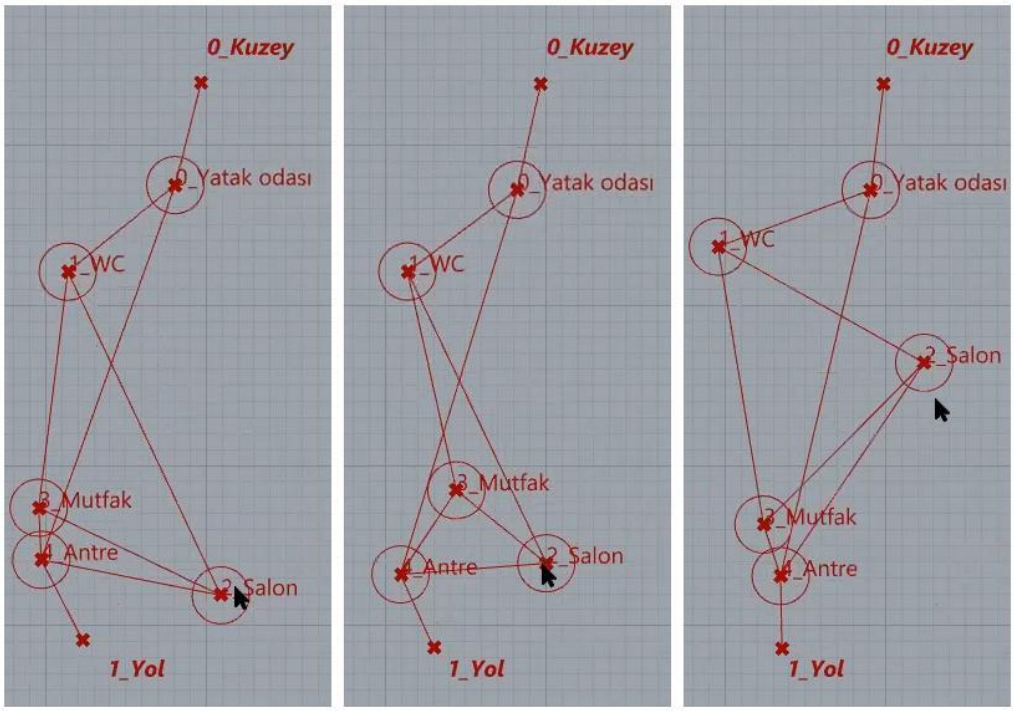


Şekil 28. *Contextual Node Manager*'in hareketli ağ üzerinde gerçek zamanlı olarak kullanılması.



Şekil 29. Link Manager arayüzü kullanılarak düğüm noktaları arasındaki ilişkilerin daha hassas biçimde kontrol edilmesi.

Benzer şekilde *Link Manager* isimli pencere de ağ hareket halindeyken kullanılarak farklı olasılıkların araştırılmasına yardımcı olacaktır (Şekil 29). Dinamik ağın hareketlerini daha anlaşılır hale getirmek için SpaceChase sekmesinde *Display* bileşen grubu altındaki *Dynamic Graph Display* isimli bileşen kullanılmalıdır. Böylelikle Kangaroo'nun işlediği hareket bilgisi, SpaceChase'in Node bilgisi ile bütünleştirilmektedir. Bu bileşen farklı görselleştirme biçimlerine olanak sağlamak üzere bütün çıktılarını Grasshopper nesneleri biçiminde sunmaktadır.

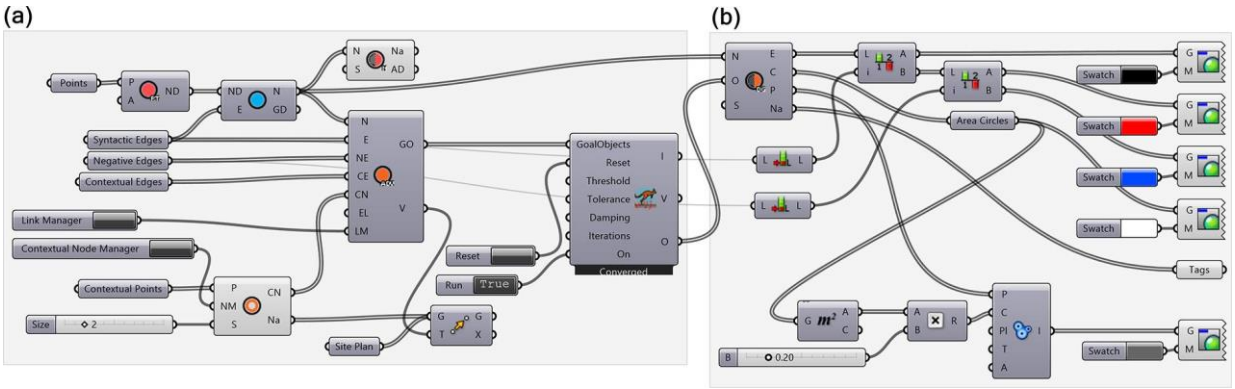


Şekil 30. Hareketli ağ yapısına elle müdahale etmek mümkündür. Bunun için fare ile düğüm noktaları tutulup sürüklenmelidir.

SpaceChase for Grasshopper ile üretilen dinamik ağların doğrudan kullanıcı etkileşimine açık diğer özellikleri ise fare yardımıyla gerçekleştirilen etkileşimler ve ağ yapısı üzerinde gerçekleştirilen güncellemeler bağlamında ele alınabilir. Kangaroo altyapısının SpaceChase'e sağladığı önemli ve özgün bir yetenek, hareket esnasında düğüm noktalarının fare yardımıyla sürüklenmesine olanak sağlamasıdır (Şekil 30).

Şekil 31'de görüldüğü gibi, SpaceChase for Grasshopper'ın son sürümü veri akış diyagramının rafine olduğu son aşamayı temsil etmektedir. Bu örnek veri akış diyagramı, dinamik bir ağ modellemektedir ve iki ana gruptan oluşur. Birinci grup (Şekil 31-a) dinamik ağın kurulumun yapılıp simüle edildiği bileşenleri içermektedir. Şekil 31-b'deki *Dynamic Graph Display* bileşeninin görselleştirdiği bağlantılar, tüm itme ve çekme bağlantılarını içermektedir. Bu bağlantıları ayırt etmek ve Grasshopper'da farklı biçimlerde görselleştirmek için bileşenin E

çıktısından elde edilen verinin ayrıklaştırılması gerekmektedir. Bu çıktıdaki veri sıralı biçimde sunulmuştur ve bu sıralama, pozitif bağlantılar (sentatktik hesaplamalara dahil edilen ilişkiler), negatif bağlantılar ve bağlamsal bağlantılar (bağlamsal düğüm noktaları ile ilişkiler) biçimindedir. Bu veriyi ayıran bir Grasshopper kurulumu yapılarak bu bağlantıların farklı biçimlerde görselleştirilmeleri sağlanabilir. Bu ayrımın ardından bağlantılar ve düğüm noktaları *Custom Preview* veya *Text Tag 3D* gibi Grasshopper bileşenleri kullanılarak görselleştirilebilir.



Şekil 31. SpaceChase örnek dinamik ağ kurulumu. Bu diyagram daha detaylı olarak EK 10'da sunulmuştur.

4.1.4 Sınırlar ve Olanaklar

Bu bölümde SpaceChase for Grasshopper eklentisinde bulunmayan, ileri geliştirme konuları olarak tespit edilen hususlar özetlenmiştir:

- **Düğümlerin akışkanlığı ve dönüşümü:** Dinamik ağ yapılarını görselleştirirken, birbirlerine yakın düğüm noktalarının birleşerek tek bir düğüm gibi hareket etmeleri, tartışılan bir konu olmuştur. Bu amaçla prototip çalışmalarından başlayarak *Metaball* adı verilen akışkan yüzeyler kullanılmıştır. Fakat, Grasshopper'ın *Metaball* bileşenlerinin arka planındaki algoritmanın tam olarak kontrol edilememesi söz konusudur. Bu sadece bir görselleştirme sınırı değil, aynı zamanda düğüm noktalarının davranışı ve süreç içerisindeki dönüşümü ile ilgili algoritmik bir sınırlamadır. İleri araştırmalarda düğüm noktalarının baştan sona sabit kalmadığı ve birleşip dönüşebildiği bir yapının en baştan tasarlanması ve bu yapının görselleştirilmesinde özgün ve yeni metaball türü hesaplamaların kullanılması mümkündür.

- **Hareket sonrası ağ hesaplamaları:** Eklenti gelişim sürecinde üzerinde sıklıkla tartışılan bir başka konu da Mekan Dizimi metriklerinin dinamik ağ yapısı üzerindeki olası etkileridir. Bu konuda net bir sonuca ulaşılamamış olmakla beraber, eklentinin son sürümünde *Syntactic* ağ modeli kullanılarak hareketin seçilen herhangi bir metrik ile ilişkilendirilmesi mümkün hale getirilmiştir. Fakat halen tanımlanan ilk ağ topolojisi sistem tarafından dönüştürülemediği için,

Mekan Dizimi metriklerinin hareket esnasında değişmesi ve ağ yapısı ile iki yönlü bir etkileşime girmesi mümkün değildir. İleri çalışmalarda hareketli ağ yapısının düğümleri birleştirip ayırdığı veya bağlantıları koparıp yeniden bağladığı durumların da simüle edilebileceği öngörülmüştür. Böylelikle Mekan Dizimi hesaplamalarının süreç içerisinde daha etkin roller üstlenmesi söz konusu olabilir.

- **"Üçüncü boyut":** İlk prototiplerinden itibaren, deneme süreçlerinde ifade edilen bir konu, çalışmanın ürettiği ağ yapılarının iki boyutlu olmasıdır. Ağ temsili, özünde boyutlardan bağımsız bir ilişkiselliği ifade ettiği için, bunun n boyutlu ortamlarda temsil edilmesi ile ilgili herhangi matematiksel sınırlama bulunmasa da kullanıcı deneyimi göz önüne alındığında düğüm noktaları arasındaki ilişkilerin durağan ağ yapılarında en rahat okunabildiği ortamın iki boyutlu düzlemler olduğu öngörülmüştür. Fakat bu durum, kurulan bağlantıların üretebileceği olasılıkların deneyimlendiği dinamik ağ ortamında olasılıkların kısıtlanmasına neden olmaktadır. Mekanların üç boyutlu ilişkiler içerisinde bulundukları senaryolarda kullanılmaya çalışılan çözüm (*workaround*), "Merdiven" gibi bir düşey dolaşım elemanının düğüm noktası olarak tanımlanması olmuştur. Fakat düğüm noktaları aynı düzlem üzerinde bulundukları müddetçe birbirlerini etkilediklerinden, ilişkilerin çok katlı mekansal sistemlerde kurulabilmesi zorlaşmaktadır. Ayrıca özellikle dinamik ağ görselleştirmesinde ekranda aynı anda bulunabilecek düğüm noktası sayısında teorik veya teknik bir sınırlama olmamakla beraber, kullanıcı deneyimi açısından bu sayının bir sınırının olduğu görülmektedir. İleri çalışmalarda, düğüm noktalarına bazı yeni parametrelerin eklenmesi ve bu yolla karmaşık üç boyutlu sistemlerin çözümü için daha kullanışlı bir aracın oluşturulması mümkündür. Bu parametreler etiket (tag) biçiminde olabileceği gibi, sayısal değer biçiminde de olabilirler.

- **Özgün simülasyon motoru:** SpaceChase for Grasshopper, Kangaroo geometrik optimizasyon motorunu amacı dışında farklı bir şekilde kullanmaktadır. Grasshopper'ın mevcut sürümünde dinamik ağ yapılarını modelleyebilecek en (ve muhtemelen tek) etkin yöntemin bu olduğu değerlendirilmektedir. Kangaroo dışında başka optimizasyon araçlarının (*Flexhopper* gibi) kullanılması da mümkündür. Bu çalışmanın sonucunda görülen bir ileri çalışma, Kangaroo'yu iş akışından çıkartmak üzere SpaceChase'e özel ve özgün simülasyon bileşenlerinin oluşturulmasıdır. Bu geliştirme çok daha derin bir araştırma ve yazılım desteği gerektireceği için bu proje kapsamında gerçekleştirilmemiştir. İleri bir çalışmada böyle bir yeteneğin eklenmesi ile SpaceChase'in sentaktik ağları harekete geçirme konusunda çok daha gelişmiş kontrollere sahip olacağını söylemek mümkündür. Bu çalışmanın aynı zamanda Grasshopper ve Rhinoceros dışında da çalışabilen bağımsız (*stand alone*) bir yazılım çalışmasının başlangıcı olacağı öngörülebilir.

- **Makine öğrenmesi:** Bu çalışmanın tartışmaya açtığı ileri bir konu ise, geliştirilen araçlarla oluşturulan ağ yapılarının gelecekte üretebileceği büyük veri ile ilgilidir. Çalışmanın başından itibaren analitik ve gözlem amaçlı olarak geliştirilen SpaceChase eklentisi, ağ verisini işleme ve kaydetme özellikleri bakımından bu tür çalışmalara öncü olabilecek özelliklere sahiptir. Mekan Dizimi'nin temsil ortamı olan ağ yapıları ile makine öğrenmesinin (*machine learning)* temsil ortamı olan yapay sinir ağları arasındaki benzerlikler kayda değerdir. Gelecekte mekansal ağ yapılarını makine öğrenmesi yöntemleriyle analiz ederek sonuçlar üretmenin mümkün olduğu değerlendirilmektedir.

4.2 SpaceChase Oyunu

SpaceChase ilişkisel mekansal konfigürasyon tasarımı süreçlerinden yola çıkan, mimarlarla birlikte genel kullanıcıya da hitap edecek oyunsal bir yazılımdır. Mekansal parçaları düğüm noktaları aracılığıyla temsil eden ve aralarındaki ilişkileri kurup bozarak olası bir araya gelmeleri araştıran SpaceChase, arka planda tüm olasılıkları ilgili Mekan Dizimi metrikleri ile ölçülebilir hale getirmektedir.

Kuramsal alt yapısı Mekan Dizimi araştırmaları üzerine kurulu olan SpaceChase oyununun tasarım çalışmaları, geçmişten ilişkisel ağ bazlı örnek tasarım süreçleri ve örnek mimari yapıların mekansal ağ modelleri incelenerek başlatılmıştır.

Örneğin, geçmişteki teorik/pratik örneklerden Frankfurt Mutfağı ile mimar Lihotzky tarafından yürütülen sayısız zaman-hareket etütleri ile mutfak dizilimlerinde verimlilik hedeflenmiştir. Dönemin düşünce biçimleri olan Taylorism ve Fordism'den yola çıkan bu dizilim-hareket ilişkisini inceleyen ekonomist Christine Frederick domestik yaşamda beden hareketleri ve sirkülasyon farklılıklarını artiküle ederek "iyi ve kötü" dizilim örneklerini 1922'de domestik hayatın rasyonalizasyonu başlığı ile yayınlamıştır (Corrodi, 2015). Frankfurt Mutfağı, mutfak içindeki ve mutfakla özdeşleşen işlev ve mekansal birimlerin konuttaki diğer birimlerle ilişkilerini deşifre ederken bir yandan da modernitenin kavramsal arka planı ile mekansal düzenleri arasındaki ilişkinin varlığını sorgulamaktadır (Mcgaughey, 2017). Mekansal ilişkilerin bu tür süreçler içinde araştırılması, proje kapsamında geliştirilen oyun için de ilham verici olmuştur.

Frankfurt Mutfağı örneğindeki gibi kavramsal düşüncenin işlevsel ve mekansal karşılıklarının aranması sürecine benzer biçimde oyun da hikaye ve sahneler üzerinden oyuncuların mekansal bağları kurmasına ve işlevselleştirmesine olanak sağlamaktadır. Sonuç olarak mimari programa atıfla farklı sahneler (konut, restoran gibi) aracılığıyla çeşitli işlevlerin ve farklı kullanıcı senaryolarının izleğinde, insan hareketlerinin ve görülebilirlik/girilebilirlik ilişkilerinin oluşturduğu

koreografinin soyut temsilleri ile oyuncuların önceden verilen ağları dönüştürerek ya da yeni ağ modelleri üreterek oynayabilecekleri şekilde bir oyun tasarlanmıştır. Öte yandan tasarım deneyimi bir çeşit *"puzzle"* yani yapboz olarak ele alınmış, mekansal kurgunun davranışsal, algısal ve sosyal karşılıklarının ifade edilebildiği, Mekan Dizimi kriterlerinin gömülü olabileceği farklı mekansal kurguların neler olabileceği etüd edilmiştir.

Oyun tasarım ekibinde, araştırmacıların yanı sıra görsel iletişim ve tasarım konusunda *Eyes Don't Lie* firması, *Unity* bazlı oyun yazılımı konusunda *Yokyer Bilişim Yazılım Araştırma ve Geliştirme Sanayi ve Ticaret Eğitim ve Danışmanlık Ltd Şti.*, oyun sahne ve senaryoların üretilmesi ve düzenlenmesinde ise *Burhanettin Emre Kuzuoğlu* ile çalışılmıştır.

Sonuçta geliştirilen ve son kullanıcıya ulaşan oyun, bilgisayarlar, konsollar ve mobil cihazlar için video oyunları ve simülasyonları geliştirmek için kullanılan *Unity* oyun motoru üzerinden üretilmiştir. Oyun, Apple iOS, Google Android mobil platformlarda ve WebGL aracılığı ile masaüstü platformlarda kullanılabilmektedir.

Oyun bir yanda araştırmacılar tarafından bir yanda davet edilen danışmanlar bir yanda da proje danışmanı tarafından test edilerek yorumlanmıştır. Bu raporun yayınlandığı tarihte de web sitesi üzerinden (www.spacechase.app) ulaşıma ve test edilmeye açıktır. Oynanan seanslar anonim olarak kaydedilmektedir; böylece yazılımda ve oyun kurgusunda geliştirme olasılıkları araştırılmaktadır.

4.2.1 Hedefler / Adımlar

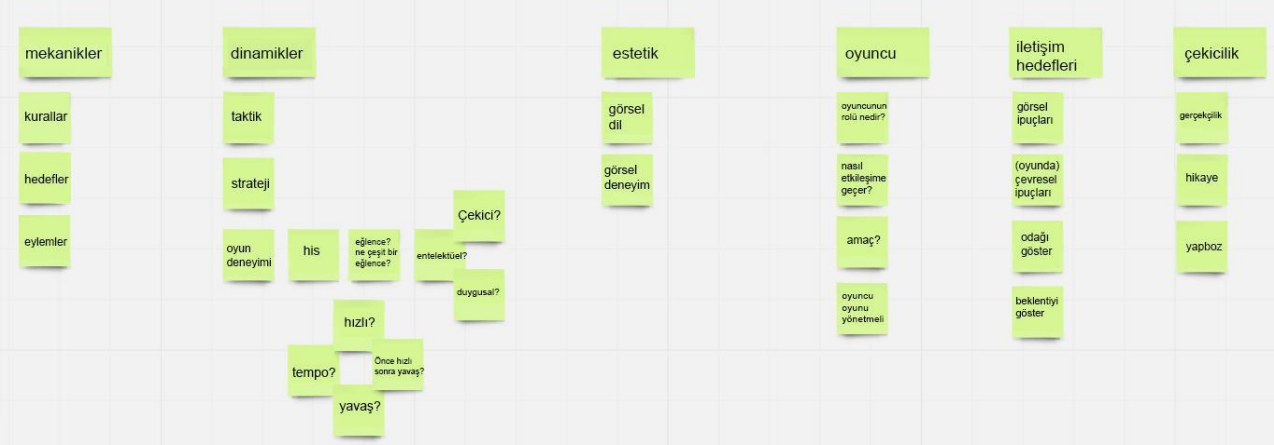
SpaceChase oyun yazılımı geliştirme sürecinin hedefleri süreç boyunca birbirleri ile etkileşimli olarak gelişen üç alt iş paketi tanımlamaktadır:

- **Oyun sahne ve hikayelerinin tasarlanması:** Bu iş paketi kapsamında, oyunun türü, formatı, bölümleri, bölüm hedefleri, oyuncu motivasyonu, oyun sahne ve seviyelenmesi görsel arayüz tasarımcısı ve araştırmacılarla metin yazarının yer aldığı ekip tarafından farklı oturumlarda tartışılmış, prototiplerle test edilmiş ardından atölyelerde deneyimlenmiştir.
- **Görsel iletişim ve arayüz tasarımı**: Bu kapsamda konsept ve görsel tasarım, oyun elemanları, oyunun temel kuralları ve mekanikleri, bölüm yapıları ve bölüm tasarımı, oyun ekranları, arayüz deneyimi ve arayüz elemanları konularında üretimler yer almaktadır. Proje ekibi ve oyun tasarımı ekibinin bir araya gelerek verdiği kararlar ve bu kararlar doğrultusunda ortaya çıkan tasarım, süreç boyunca tüm paydaşların geri dönüşleri ile beslenerek güncel halini almıştır. Bu kapsamda tasarım bileşenleri oyun yazılımına aktarılabilecek formatta çıktılarla teslim edilmiştir.

- **Unity bazlı oyun yazılımının geliştirilmesi.** Oyun türü ve mekaniklerinin tariflenmesinin ardından Unity'de kütle çekim prensibine dayalı ağ modellemesi kullanılarak oyun yazılımı geliştirilmiştir. Bu iş paketi kapsamında oyun ortamının programlanması, oyuncu arayüzü tasarımlarının yazılıma aktarılması, yönetici (admin) arayüzlerinin kodlanması ve ilişkilendirilmesi, Mekan Dizimi metriklerinin hesaplanması ve bilgilendirme arayüzü ile okutulması, oyun seanslarının kaydedilmesi, beta testler yapılarak deneyime dayalı geri beslemelerin programa işlenmesi, kullanım kılavuzu kaydedilmesi işleri tamamlanmıştır.

4.2.2 Geliştirme Süreci

SpaceChase Oyunu'nun, mimari tasarım kuram ve pratiğinden gelen ağ teorisi ve Mekan Dizimi üzerine geliştirilen bir dijital bir oyun olması hedeflenmiştir. Bu yüzden tasarım süreci, ilgili kuramsal birikimin mimarlık pratiğinin dışından gelen tasarım paydaşlarına aktarılması ve irdelenmesi ile başlamıştır. Görsel tasarım ve yazılım ekibi oyunun mantığını oluşturan ağ teorisi ve Mekan Dizimi hakkında bilgi sahibi olurken araştırma ekibindeki mimarlar da oyun mekanikleri ve türleri ile ilgili bilgi edinmişler ve Mekan Dizimi araştırmalarının bir oyuna nasıl dönüştürülebileceğine ilişkin yöntem geliştirme çalışmalarını gerçekleştirmişlerdir (Şekil 32).



Şekil 32. Oyun tasarım alanının temel alt başlıkları.

Bu noktadan itibaren oyun geliştirme sürecinin şu adımlarla ilerlediği söylenebilir: (1) çalışma için ilham verici olabileceği düşünülen mevcut oyun yazılımların incelenmesi ve oyunlaştırma kalitelerinin araştırılması, (2) oyun konsept ve görsel tasarımının geliştirilmesi, (3) yazılım geliştirme çalışmaları: sistem tasarımı, (4) uygulama ön araştırmasının yapılması, (5) prototip uygulama çalışmaları.

Oyunlaştırma süreci, mevcut oyun yazılımlarının incelenmesi sonucunda SpaceChase oyununun amacının kurgulanması, bölüm yapıları ve hedeflerinin tasarlanması, oyuncunun

oynama ve başarı motivasyonlarının belirlenmesi konularını kapsamaktadır. Bu süreçte yapboz (puzzle) çözümü ve ağ modellemesinin etkin olduğu farklı “oyun” türleri ve oyunlar incelenmiş, örnek alınabilecek kaliteler ve hedeflenecek farklılıklar çıkarımlanmıştır (Şekil 33). Örnek olarak incelenen oyunlar arasında gramer tabanlı bir oyun olan *Baba is You,* oyunun kurallarının oynarken değiştirilebildiği bir bulmaca oyunudur. Her seviyede, kurallar etkileşime girebilecek bloklar olarak mevcuttur; oyuncu bunları manipüle ederek seviyenin çalışma şeklini değiştirebilir ve şaşırtıcı, beklenmedik etkileşimlere neden olabilir. Tasarım sürecine benzeşen bu kalitenin SpaceChase oyununun farklı seviyeleri ya da farklı oyuncu modlarında da olabilmesi planlanmıştır (Bkz. yönetici ekranı, Bölüm 4.2.3).

Mini Metro oyun örneği ise konumlar arasında bağlar kurarak akışın planlanması üzerine kurulu bir ulaşım ağı simülasyon oyunudur. SpaceChase oyunu bağlamsal düğüm noktalarının sabit, mekansal ve işlevsel düğüm noktalarının ise dinamik olması sebebiyle bu oyundan farklılaşmaktadır.

Planarity oyunu ise ağ modelinde bağların kesişmesini engellemek üzerine kurulu topolojik bir oyundur. Burada çek-bırak yöntemi ile düğümlerin yeri değiştirilerek bağların kesişimi önlenmektedir. Mekansal ağlarda bağ kesişimlerinin yeni düğümlere dönüşebileceği düşüncesinden dolayı incelenmiş, ancak SpaceChase oyununda düğüm noktası yerlerinin ve hareketlerinin diğer düğümlerle yakınlaşma ve uzaklaşma eğilimi şeklinde belirlenmesine karar verilmiştir.

Bu aşamada incelenen diğer oyunlar, *Osmos, Olo, Agario* gibi düğüm noktası temelli evren kurgulu soyut oyunlar, *Mindustry, Bridge Constructor, Factorio* gibi simülasyon oyunları ile *Opus Magnum* gibi kural bazlı yapboz oyunlarıdır. Bir hedefe ulaşmanın birçok yolu olan açık alanlar olarak tasarlanmış “açık dünya” oyunları incelenerek SpaceChase oyunun doğrusal olmayan açık uçlu geliştirilme olasılıkları tartışılmıştır.



Şekil 33. Yapboz çözümü ve ağ modellemesinin etkin olduğu var olan oyun kaynaklarının incelenmesi.

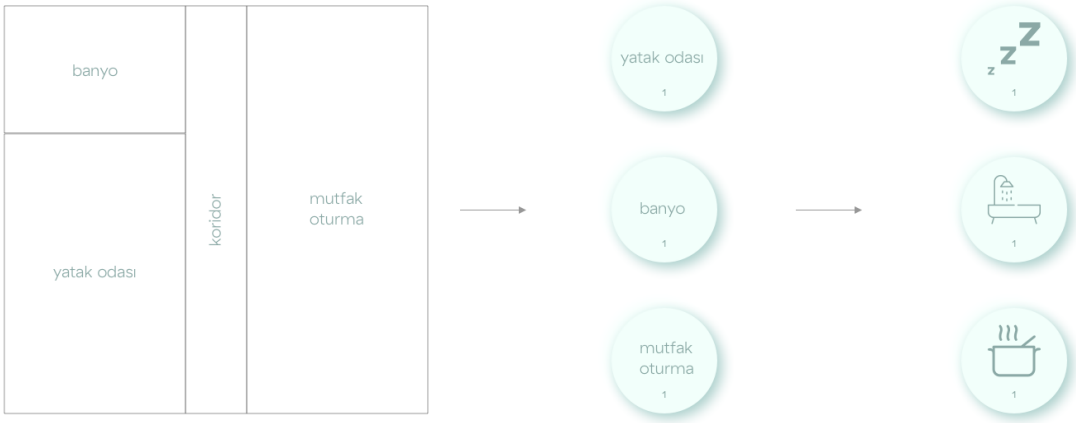
Mimarlık pratiği ile oyun tasarımı ve oyun yazılımı pratiklerinin bilgi ve deneyimlerinin paylaşıldığı ön araştırma süreci, “oyun”, “oyunsallaştırılmış tasarım” ve “tasarım süreci” arasındaki benzerlik ve farklılıkları tartışmayı gerektirmiştir.

“Oyun” kavram olarak mimarlar tarafından ucu açık sonuç odaklı olmayan bir model gibi algılanabiliyorken oyun tasarımcıları tarafından bir spor oyunu gibi net kuralları olan veya skor/başarı gibi bir hedefi olan bir ortam olarak ele alınmıştır. Öte yandan kuralları aynı olmasına rağmen her oyunun basketbol örneğindeki gibi farklı koreografilerle gerçekleşmesi, tasarım tarafında mekansal örüntüler ve onların farklı hallerine benzetilmiştir. Sonuç olarak *Unity* ortamında hazırlanan yazılımın sadece mimarlık öğrencilerine değil, ilgili herkese hitap edebilmesi, mekansal kalite ve Mekan Dizimi kriterlerini anlamaya ve öğrenmeye fırsat yaratabilecek özelliklere sahip olması hedeflenmiştir. Dolayısıyla Oyun’da mimari tasarım süreci birebir simüle edilmez; oyuncu mimar olmayabilir ancak tercihlerini yaparken mekansal kurgunun etkilendiğini görmektedir. Oyuncunun, mekansal düzenlerin oluşumuna yönelik insan ihtiyaçları ve tercihleri ile şekillenen dizilimlerin mimari karşılıklarını gözlemleyebilmesi istenmiştir.

Oyun kurallarının tasarlanmasına paralel yukarıda sözü geçen koreografilerin yaratılmasında önsel/ilksel (apriori) ilişkisel bir matris, yani olasılıklar bilgisinin verilmesine karar verilmiştir. Bu bilgi, bir çözük örüntü gibi mekansal bileşenler arasından gevşek bir ağ ilişkisini tarif etmekte, oyuncunun hamleleri bunları kuvvetlendirmekte ya da yok etmektedir. Oyun böylece iki ekrana bölünmüştür; yönetici ekranı ve oyun ekranı. Yönetici (admin) ekranında girilen bu ön ilişkisel harita oyunun kurulu olduğu şablonu belirleyen bir ön tasarım alanıdır. Bu "yürütücü ekranı" mimarlarla gerçekleşen workshoplarda tanıtılmıştır ve mimar oyunculara gelecekte de açık olması planlanmıştır. Yönetici ekranına giriş yapan biri, bir yanda oyunu tasarlarken bir yandan da oynayanların verdiği tepkilerle olası farklı karşılıkları gözlemleyebilmektedir.

Oyun türü ve mekaniklerinin tariflenmesi aşamasında, var olan mekansal kurgu eskizleri tartışılmış ve olasılıklar simüle edilmiştir. Öte yandan Unity'de kütle çekim prensibine dayalı ağ modelleri algoritması yazılmıştır. Prototip oyun senaryolarının yazılması, tasarım ekibinin bu senaryoları simüle edip geliştirmesine paralel, genel grafik dilin tasarlanması ve bileşenlerin tasarımlarının araştırma ekibine sunumu gibi adımlar, eş zamanlı olarak gelişerek oyunun tasarım sürecini oluşturmuştur. Bu süreç *Miro* çevrimiçi ortak beyaz tahta tasarım arayüzünde ve uzaktan toplantı ortamlarında yürütülmüştür.

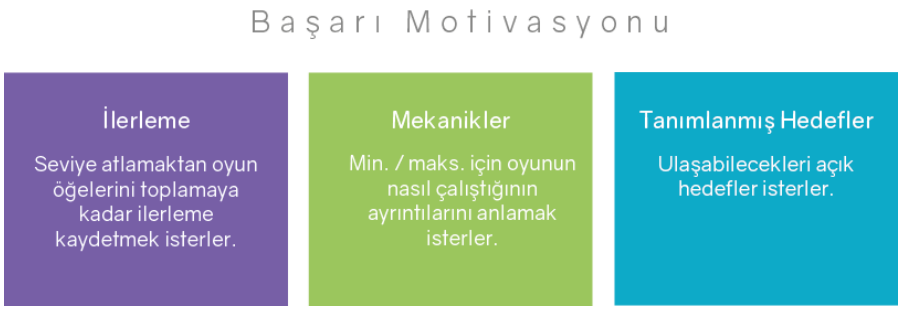
*Oyuncunun amacı***,** her bölümde bir hikaye çerçevesinde verilen işlevleri (yemek, yatmak, temizlik gibi) yerine getirecek şekilde arayüzde verilen mekanları oyun alanı içinde bağlayarak dizmek/düzenlemek olarak belirlenmiştir. Mekansal bileşenlerin soyut bir şekilde gösterilmesi ve aralarında ilişkisel bağların kurulması bu dizim ve düzenlemenin temel özellikleridir. Mekanlar, istenirse isimleriyle değil (örneğin; yatak odası) ifade ettikleri fonksiyonlarla (örneğin; uyku) da betimlenebilmektedir. Birbirleriyle fonksiyon açısından ilişkilendirilen mekanlar bir araya gelebilir ve tek bir mekanı ya da mekansal kümeyi ifade edebilecek şekilde dizilebilmektedir (Şekil 34).



Şekil 34. İşlevleri ile tanımlanmış mekansal birim örnekleri.

Oyun mekaniğinde, sahne temaları (kırda konaklama, sitede kreş, gibi) *bölümlere ayrılmış ve her bölümün amaçları/hedefleri* bir hikaye etrafında kurgulanan, sırasız ilerleyen, ağ teorisi ve Mekan Dizimi ile desteklenebilen direktifler çerçevesinde kurgulanmıştır. Zorluk seviyesi düşük bir bölüm oyuncudan "bir dere kenarında konaklama" yapmasını / tasarlamasını isterken, daha ileri zorluk seviyesindeki bir bölüm ise oyuncudan "yemek ve çalışmanın aynı mekanda olmasını ve uyuma dışındaki işlevlerin bağlamsal düğüm noktalarından kolaylıkla ulaşılabilir olmasını" isteyebilmektedir. Dolayısıyla, basit olan "konaklama" hedefli bölüm sadece dere kıyısı-kamp ilişkisinin kurulması ile tamamlanırken, zor olan bölümde oyuncunun mekanların bağlarını kurup çözerek Mekan Dizimi değerlerini de kontrol etmesi gerekmekte ve bu oyuncunun başarısını etkilemektedir.

Oyuncunun oynama motivasyonu anlık olarak verilen hedeflere karşılık gelen ağı çözmek, genel olarak işlevsel ve mekansal dizgiler üzerine düşünmek ve özel olarak da Mekan Dizimi ve mekansal ağ düşüncesini anlamak/öğrenmek ve pratikte kullanarak kendini geliştirmek olarak belirlenmiştir. Ayrıca oyuncuya mekan ilişkilerini kurgularken özgürlük verebilmek için bölümlerin tek geçerli çözümünün üretilmesi değil, farklı dizimlerle de geçilebilmesi hedeflenmiştir (Şekil 35).

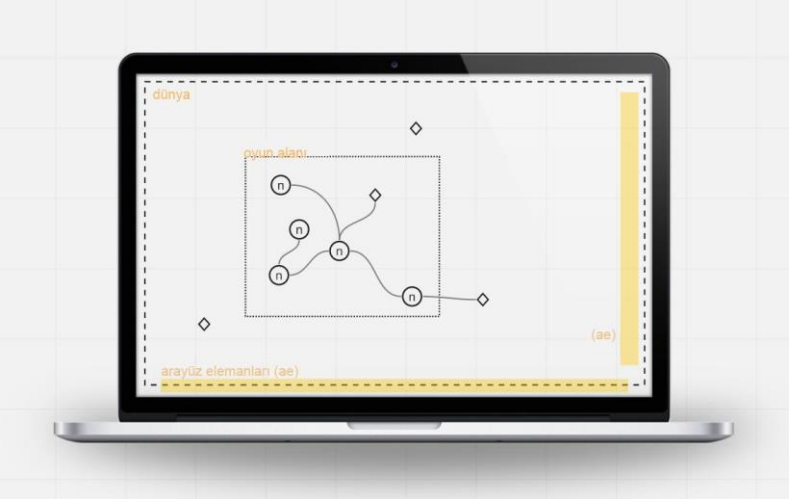


Şekil 35. Farklı oyun mantıklarından oyuncu başarı motivasyonu karakteristikleri.

Oyun Konsept ve Görsel Tasarımı özetle arayüz ön tasarımı, arayüz etkileşim tasarımı ve oyun elemanlarının tasarımı iş kalemlerini kapsamaktadır.

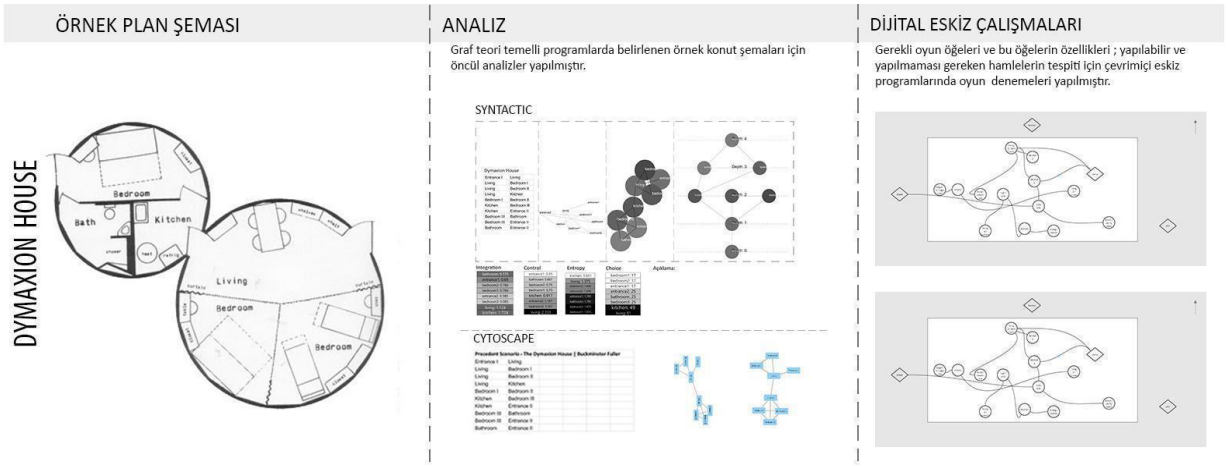
SpaceChase Oyunu, tut ve çek (*point and drag*), senaryo tabanlı yapboz türü bir mekansal dizilim oyunudur; oyuncunun amacı her bölümdeki mekansal hikayeyi farklı olasılıklar dahilinde kurgulamaktır. SpaceChase'in simülasyon oyunlarından farkı, mekansal öğelerin temsillerinin (işlev, oda ya da mekansal karakter gösteren odaklar) fizik kuralları dahilinde çoklu seçeneklerle eşlenebildiği bir arayüzle oyuncuların dinamik ilişki ağları oluşturarak tasarım yapabilmelerine

olanak sağlamasıdır. Oyun, aynı zamanda oyuncunun ürettiği bu mekansal ağların kalitelerine ilişkin bilimsel (Mekan Dizimi temelli) veriler ortaya koyacak şekilde tasarlanmıştır.



Şekil 36. *Miro* dijital arayüzünde proje ekibin gerçekleştirdiği ortak eskiz örneği.

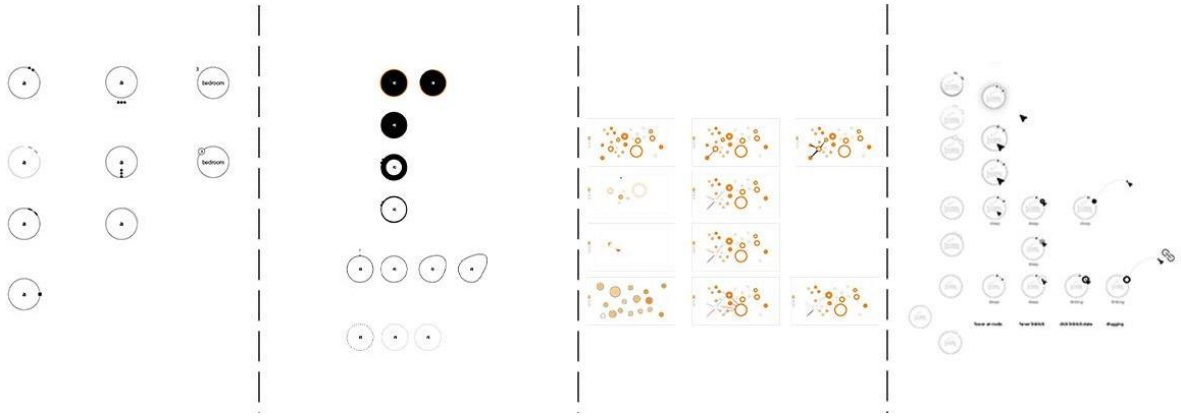
Oyunun kurallarını ve öğelerini belirlemek için çevrimiçi toplantılarda dijital eskiz yazılımları kullanılmıştır. Bu toplantılarda canlı olarak hem basit hem de karmaşık oyun senaryoları kurgulanmıştır (Şekil 36). Oyun basit görseller ve akışlardan yararlanılarak birçok defa oynanmıştır (Şekil 11). Oyunun arayüzünü oluşturacak elemanlar, bu elemanların statik, dinamik; çeken, iten, vb. olma özellikleri, oyuncunun elemanlarla yapabileceği ve yapmaması gereken hamleleri oyun eskiz halde oynanırken belirlenmiş ve oyuncunun bu arayüzle nasıl etkileşimde bulunabileceği planlanmıştır. Mekan temsilinde ise soyut bir dil tercih edilmiştir.



Şekil 37. Oyun tasarım sürecine dair üretimlerden örnekler.

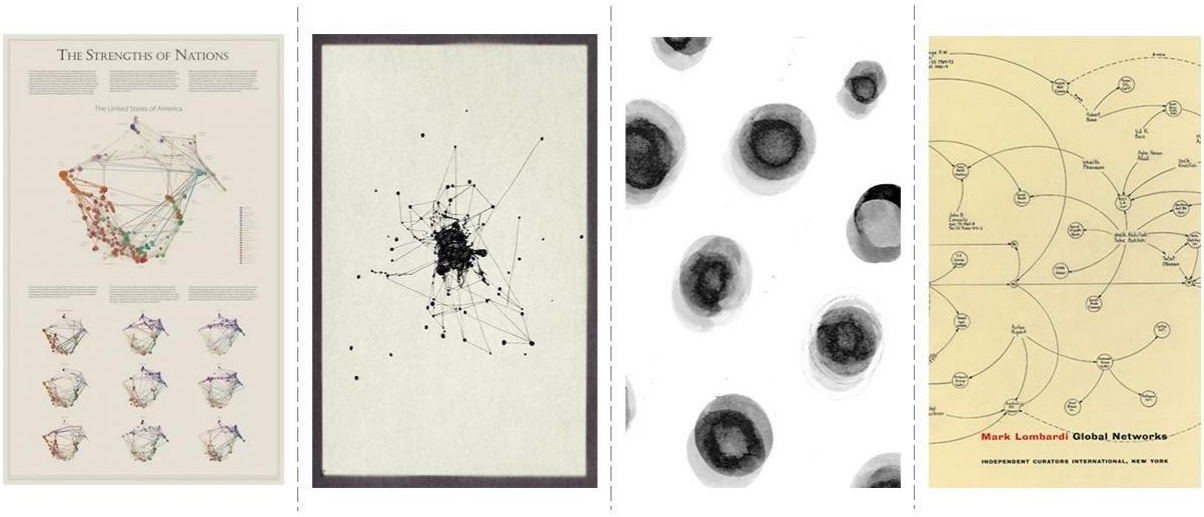
Yukarıda söz edilen çalışmalar neticesinde oyunun kuralları, oyun öğelerinin (örneğin düğüm noktaları, bağlar) özellikleri, bölümlerin ve genel oyunun akışı şekillenmiştir. *Görsel tasarım ve*

arayüz ön tasarımları kapsamında çizgisel eskizler ve farklı arayüz düzenleri geliştirilmiştir (Şekil 38).



Şekil 38. Dijital eskiz örnekleri.

Sözü edilen oyun malzemeleri netleştirildikten sonra, oyunun görsel konsept tasarımı kademeli olarak ortaya çıkmıştır. Oyun, gösterim yöntemi olarak ağ diyagramlarına odaklandığı için ağ görselleştirmelerinden yararlanılmış ve basit/organik yuvarlak şekiller esin kaynakları olarak alınmıştır (Şekil 39). Yalın bir görsel dil seçilmesinin nedeni, oyunun karmaşık yapısını olabildiğince net ve basit bir şekilde sunmak olarak özetlenebilir.

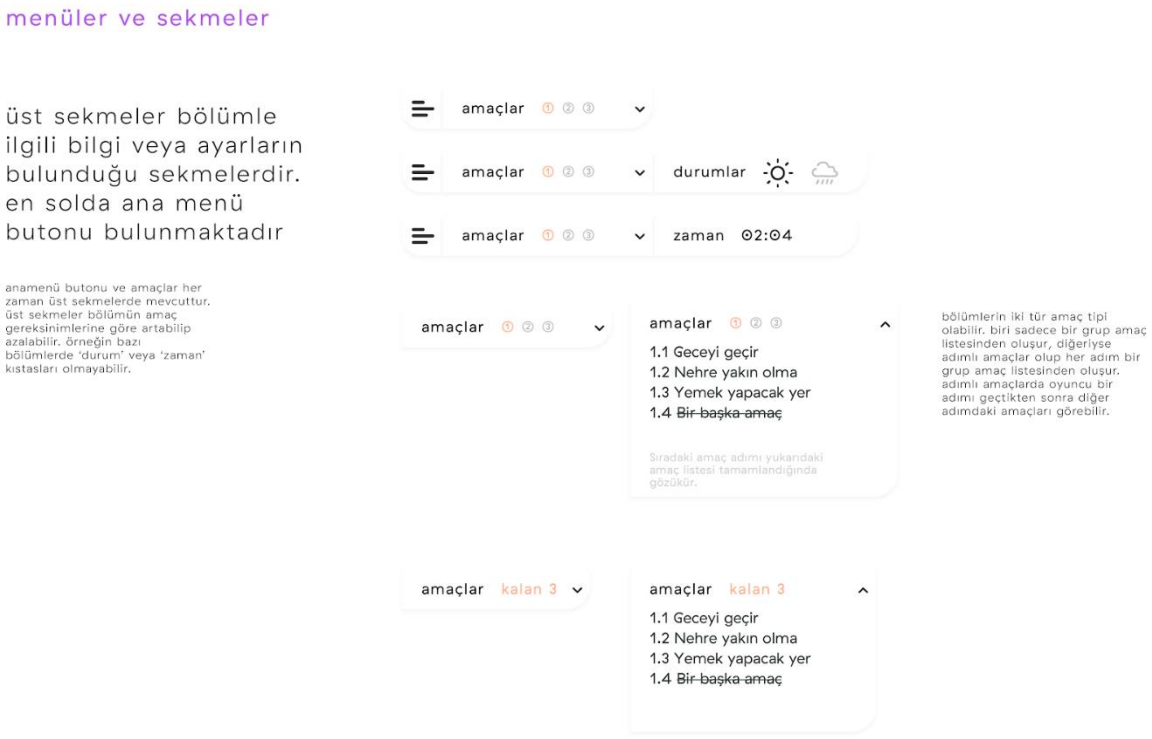


Şekil 39. Görsel esin kaynakları örnekleri.

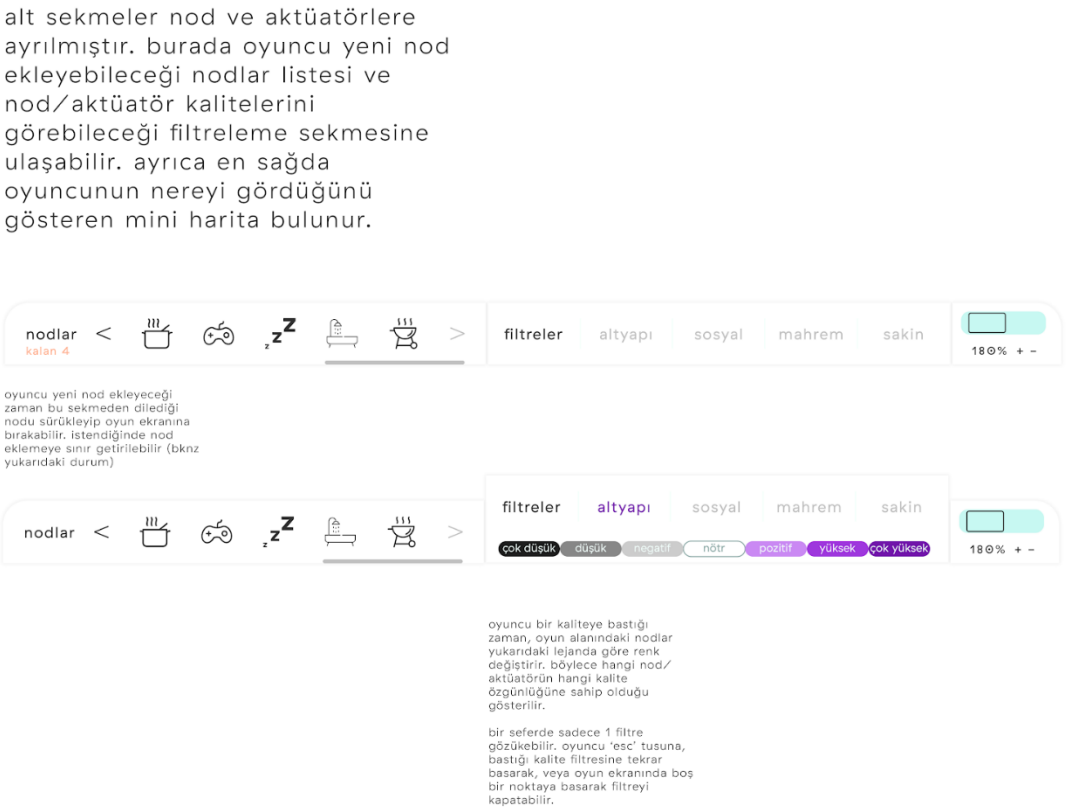
Arayüz etkileşim tasarımı bağlamında ise oyuncunun etkileşimini gerçekleştirecek bir seri iletişim, geri besleme ve farkındalık bileşenleri üzerinde çalışılmıştır. Bu çalışmaların oyun ekranındaki karşılıkları EK 13'te detaylı olarak görülebilmektedir. Oyunun süre kısıtının olması ile geri sayım bileşeni, düğüm noktalarının ikonik ifadeleri, düğüm havuzu, hava şartları gibi ortam tipleme düğmesi, düğümlerin kendi aralarındaki çekimlerle hareketlilikleri, oyun

sınırlarından çıkma uyarısı veren renk değişimleri, farklı metriklere bağlı renkle görselleştirme olanağı, düğümlere basınca bir alt seviye etkileşim alanı okunabilmesi gibi olasılıklar araştırılmış, bunların görsel tasarımları gerçekleştirilmiştir.

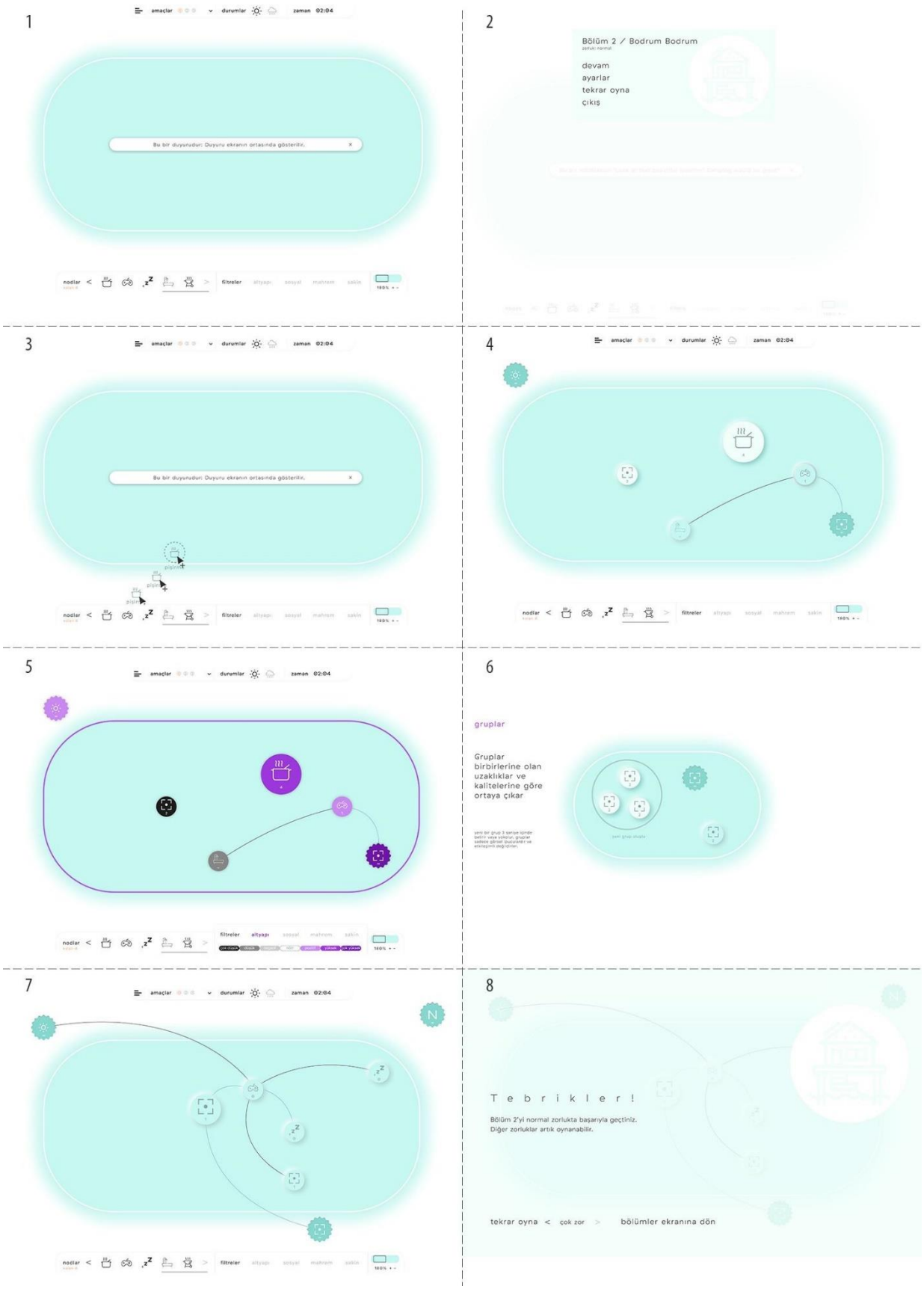
İlk prototip arayüzü iki ana bardan oluşmaktadır (Şekil 40 ve 41). Bu barlar bloklardan oluşur ve bölümdeki amaçlara göre artar veya azalır. Üstte yer alan bar, bölümün amaçlarını yansıtacak şekilde kullanılır. İçinde bölüm amaçlarının görülebildiği blok, bölümün zaman kısıtını gösteren blok ve opsiyonel olarak da bölüm durumlarının (hava durumu, ya da iklim, ortam gibi modlar) seçilebildiği blok ile bulunur. İlerleyen oyun versiyonlarında bu bloklar arttırılabilir. Alt blok ise düğüm noktası ve düğüm noktası özelliklerine ayrılmıştır. Oyuncu buradaki blokları kullanarak oyuna yeni düğüm noktaları ekleyebilir, var olan düğüm noktası/bağlamsal düğüm noktası kalitelerine göre filtreli görebilir veya (eğer düğüm noktası ayarlama özelliği etkinse) düğüm noktası ayarlarını değiştirebilir (Şekil 40 ve 41).



Şekil 40. Oyun arayüzünü meydana getiren iki ana bardan biri (üst blok).



Şekil 41. Oyun arayüzünü meydana getiren iki ana bardan biri (alt blok).

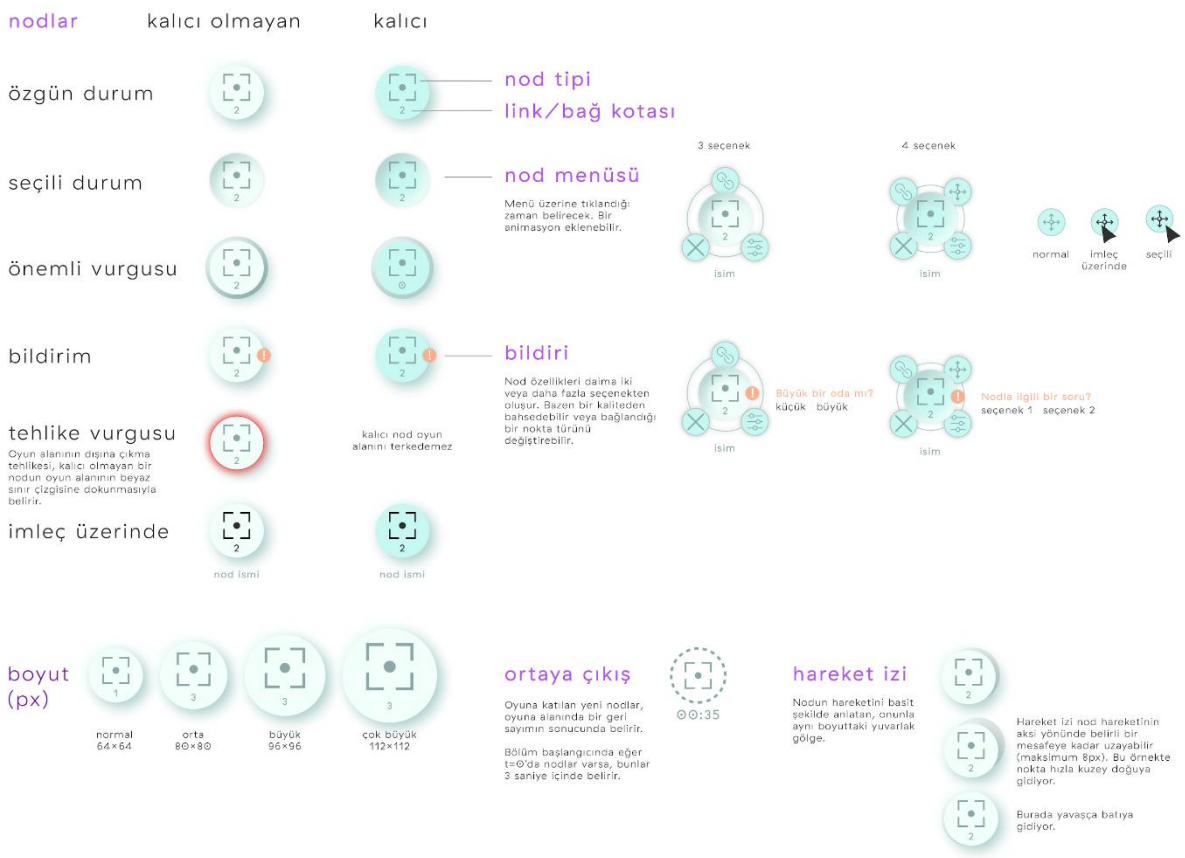


Şekil 42. Oyunu meydana getiren adımlar şeması.

SpaceChase Oyunu'ndaki en önemli etkileşim oyuncunun bağ kurmasıyla gerçekleşir. Dolayısıyla oyuncu bir düğüm noktasına fare imleciyle (masaüstü versiyonda) veya parmağıyla (mobil versiyonda) basılı tuttuğunda o düğüm noktasını bağlayabileceği bir 'ip'/çizgi belirir. Bunu kullanarak oyuncu bağlama işlemini gerçekleştirebilmektedir. Aynı zamanda, bağların üzerine gelerek silme seçeneğiyle onları yok edebilmektedir (Şekil 42).

SpaceChase mekan temsilleri ve bunların arasında kurulacak bağlardan oluştuğu için *üç temel oyun elemanı* vardır. Bu temel elemanlar özetle aşağıdaki gibi sıralanabilir:

1. Düğüm (Node): Oyunda eylemler ve bu eylemlere sahne olan mekanlar birer düğüm olarak temsil edilir. Düğümler birer mekan veya mekansal fonksiyonu gösterir. Uyuma, oyun oynama, duş yapma, pişirme, yeme, müzik dinleme vb. tüm fonksiyonlar birer düğüm noktası olarak temsil edilir (Şekil 43). Bunlar oyun alanında dinamik bir şekilde durmadan hareket ederler. Bu hareket düğümden düğüme ve bağlamsal düğüm noktasından düğüme uygulanan itme ve çekme kuvvetleriyle oluşur. Bu kuvvetler adeta bir yay/manyetik alan gibidir. İtme-çekme kuvvetleri düğüm noktalarının ve bağlamsal düğümlerin yönetici ekranından verilen değerleriyle belirlenmektedir.



Şekil 43. Düğüm çeşitleri için görsel tasarım örnekleri.

2. **Bağlamsal Düğüm (Actuator / Contextual Node):** Bir tür düğümdür ve düğümün tüm özelliklerini taşımaktadır. Bunların diğer düğüm noktalarından en büyük farkı ekranda sabit durmasıdır; özellikle bir fonksiyona işaret etmeleri gerekmez. Bağlamsal düğüm noktaları bir mekanı temsil edebileceği gibi fiziksel bir sabit veya bir olgu da olabilir; örneğin, güneş, kuzey yönü, gürültü, otoyol, çöp kutusu birer bağlamsal düğüm olarak tanımlanabilmektedir (Şekil 44).



Şekil 44. Bağlamsal düğümler (actuator) için görsel tasarım örnekleri.

3. **Bağ (Link):** Düğümler arası (düğüm-düğüm) veya bağlamsal düğüm-düğüm arası ilişkiyi sağlayan ara bağlardır. Bağlar oyuncu tarafından oluşturulur veya yok edilir. Düğümler arası bağ bir kapıyı ifade ederken, bağlamsal düğüm-düğüm arasındaki bağ bakmayı/duymayı da ifade edebilir, ulaşılabilirliği ifade etmesi gerekmez. Her düğümün bağlanma kotası bulunması oyun kuralı olarak benimsenmiştir (bir mekanın komşularının da sınırlı olabileceği gibi). Bağlamsal düğüm noktaları gerekirse sonsuz kotaya sahip olabilmektedirler. Bağlanma, düğüm ve bağlamsal düğümlerin itme-çekme kuvvetlerini değiştirmektedir. Oyun sırasında düğümler herhangi başka bir düğüme ya da bağlamsal düğüme bağlanmamış ise mekansal hikaye veya Mekan Dizimi metrikleri gereği yönetici tablosunda girilen değerle birbirlerini çekip itmektedir. Oyuncu, düğümleri birbirine bağladığında ise bu etki artar; tersi durumda daha çok itmesine de olanaklıdır. Bağların uzunluklarına sınır verilerek, itme-çekme kuvvetleri sonucu sınırın üzerine çıkıldığında bu bağların kopması sağlanmaktadır. Oyun elemanlarının kullanımına ve aralarındaki ilişkilere dair detaylı anlatıma EK 14'te yer verilmektedir.

Yazılım Geliştirme, Sistem Tasarımı sürecinde SpaceChase oyunsallaştırma servisi ve uygulamaları çok katmanlı servis mimarisi kullanılarak tasarlanmıştır. Oyun içinde kullanılacak sahneler, parametreler ve araştırmanın sonucunda analizi yapılacak oyuncu davranışlarının kayıtları veritabanına kaydedilerek saklanmaktadır. Veritabanının önünde, diğer servis ve

uygulamaların verilere ulaşmasına olanak veren ve bu servis ve uygulamalardan gelen verilerin veritabanına kayıt edilmesinden sorumlu olan bir veri erişim servisi bulunmaktadır.

“Oyuncu”nun kullanacağı uygulamanın masaüstü ve mobil versiyonları bulunmaktadır. Kullanılan cihaza göre adapte olan bu uygulamalar, masaüstü bilgisayarlarda klavye ve fare ile etkileşim sağlayacak, mobil cihazlarda ise dokunmatik ekran ile etkileşime geçilebilecek şekilde tasarlanmıştır. Farklı ekran boyutlarına göre tasarımsal varyasyonlara sahip olmalarına rağmen, prensip olarak son kullanıcılara aynı işlevsellik ve deneyimi sağlamaktadırlar. Son kullanıcıların bu uygulamalar üzerinde yaptığı eylemler yine veri iletişim servisi aracılığıyla veritabanına kaydedilmektedir.

Sistem yöneticilerinin (admin) kullanımına senaryo ve parametre yönetim arayüzü sunulmuştur. Bu arayüzü kullanarak sistem yöneticileri, uygulama içinde kullanılabilecek yeni sahneler yaratabilmekte, uygulama içinde kullanılan algoritmalar arasında seçim yapabilmekte ve parametrelerde hızlıca değişiklikler yapabilmektedirler. Bu arayüz hızlıca test ve iterasyon yapıp, sonuçları anında görüp, analiz ve iyileştirmelerin yapılmasını sağlamıştır.

Ayrıca veri analiz arayüzü, araştırmacıların, kullanıcıların (oyuncuların) yarattığı etkileşim verilerine ulaşabilmelerini sağlamaktadır. Veri analiz servisi, veri erişim servisi üzerinden bu verileri almakta, analistin kullanabileceği formata çevirmekte ve analiz arayüzüne aktarmaktadır. Bu şekilde tasarımcılar, analistler ve isteyen oyuncular uygulama içinde yapılan hareketleri inceleyebilmekte, kullanım istatistiklerine ulaşmakta ve istedikleri analiz raporlarına ulaşabilmektedirler.

Aynı zamanda, oyun süresince oluşturulan graflar oyun sonucunda veri analiz servisine yüklenmektedir. Servise yüklenen graflar üzerinden sunucu erişilebilirlik grafiklerini (Justified Graph) üretmektedir. Bu grafik, Mekan Dizimi metriklerinin hesaplanmasına olanak vermektedir. Oyun çalışmasına dahil edilmiş olan Mekan Dizimi metrikleri SpaceChase for Grasshopper’a benzer şekilde bağlılık *(connectivity),* entegrasyon *(integration),* kontrol *(control)* ve seçim *(choice)* değerleri olmaktadır.

Hesaplanan metrikler, oyuncuların ağı/örüntüyü oluşturma süreçleri, oyun sonunda oluşan graf ile birlikte saklanarak araştırmacıların kullanımına sunulmaktadır. Oyun kapsamında hesaplanan tüm metrikler dışarı aktarılabilir durumdadır; hazırlanan bu graflar, SpaceChase for Grasshopper eklentisi içine de kolayca aktarılabilecek bir formattadır. Bu şekilde, oyuncular tarafından oluşturulan verilerin SpaceChase for Grasshopper uygulaması ile de analiz edilebilmesi mümkün kılınmıştır.

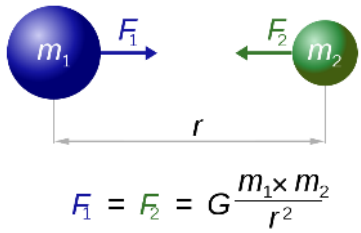
Bunun yanı sıra, Mekan Dizimi metriklerinin gerçek zamanlı olarak oyun içinde hesaplanması ve oyunun akışına etki edebilir / dönüştürebilir bir formatta olması durumu henüz gerçekleştirilememiştir. Ancak mevcut birikimle bu konunun projenin gelecek çalışmalarında öncelikli olarak ele alınabileceği düşünülmektedir.

Oyuna ilişkin *Sistem Uygulama Detayları'*na bakıldığında veritabanının, *MySQL* veritabanı yönetim servisi üzerinde yaratılmış ilişkisel bir veritabanı olduğu söylenmelidir. Veritabanı kapsamında uygulama içi sahne bilgileri, ayarlar ve ilgili parametreler, son kullanıcı bilgileri ve yaptıkları eylemler tutulmaktadır. Veri erişim servisi *Node JS* ile geliştirilmiş bir *REST* servisidir. Ana amacı diğer sistem bileşenlerine veri sağlamak ve bu bileşenlerden gelen bilgileri veritabanına kaydetmektir. Veri analiz servisi de NodeJS ile geliştirilmiş bir servistir. Üzerinde REST servisinden gelen verilerin transformasyonunu yapan fonksiyonlar barındırmaktadır. Veri analiz arayüzü, veri analiz servisinden beslenen bir *HTML5* web uygulamasıdır. Sahne ve parametre yönetim arayüzü de sistem yöneticilerinin kullandığı HTML5 web uygulamasıdır.

Son kullanıcıların kullandığı uygulamalar *Unity 3D 2019* platformu kullanılarak geliştirilmiş uygulamalardır. Unity, öncelikli olarak bilgisayarlar, konsollar ve mobil cihazlar için video oyunları ve simülasyonları geliştirmek için kullanılan ve *Unity Technologies* tarafından geliştirilen çapraz platform bir oyun motorudur (URL-2).

Apple iOS, Google Android mobil platformlarda ve WebGL aracılığı ile masaüstü platformlarda kullanılmak üzere 3 ayrı uygulama olarak çıktı alınmaktadır. Böylece kullanıcıların kullanabileceği tüm platformlar desteklenmiş olmaktadır. Proje verilerinin en kolay mobil platformlar ile toplanacağı öngörülmektedir. Fakat test ve geliştirme süreçlerinde ve atölye çalışmaları gibi organizasyonlarda en verimli kullanımın masaüstü versiyonu ile birlikte olacağı düşünülmektedir.

Uygulama Ön Araştırması'nda ise oyunda, hareketli ve hareketsiz elemanların (düğümlerin) birbirleriyle olan ilişkilerini animasyonlu bir şekilde simüle etmek üzere Newton fiziğini kullanan ve evrensel kütle çekim yasası ile elemanlar arasındaki anlık kuvvetleri hesaplayan bir yöntem geliştirilmiştir (Şekil 24). Bu hesaplamalar sonucu elemanların anlık ivmeleri bulunmakta ve bu ivmeler, elemanların hızlarını değiştirmektedir. Sonuç olarak, düğümler arasında, gezegenlerin birbirlerinin çekim alanlarının etkisinde kalarak yörüngelerde hareket etmelerine benzeyen, itme-çekme kuvvetlerini barındıran bir animasyon oluşturulmaktadır.



Şekil 45. Newton'ın evrensel kütleçekimi yasasının formülasyonu.

Prototip Uygulama aşamasında, en kısa sürede sahne yaratımının başlaması, algoritma ve parametre testlerinin yapılabilmesi için öncelikli olarak prototip bir sistem ve uygulama üretimi yapılmıştır. Bu kapsamda grafik tasarımı olmadan prototip bir masaüstü uygulama yaratılmış, bu uygulama için olabilecek en basit veri erişim servisi yaratılmıştır (Şekil 46). Klasik bir veritabanı yerine *Google Sheets* altyapısı ile bir veri tasarımı yapılmış, böylece yönetim arayüzlerine gerek kalmadan sahne ve ayarların yapılmasına ve hızlıca test edilmesine olanak sağlanmıştır (Şekil 47) (Tablo 3-4).



Şekil 46. Prototip sistem akış diyagramı.

Tablo 3: Sahneler için parametre ayarları.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	G =	0.25				İtme ve çekme çarpanları: Yeşil alanlar, bağlı olma durumunda, turuncu alanlar bağlantı olmadığında kullanılacak. -100% ile +100% arasında.																
2	**Node**	**Kütle**	**Link K.**	**Ctnx**		C1	C2	C3	C4	N01	N02	N03	N04	N05	N06	N07	N08	N09	N10	N11	N12	
3	C1	50	10	☑	C1	**C1**	0%	0%	0%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	C1
4	C2	50	10	☑	C2	0%	**C2**	0%	0%	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%	C2
5	C3	50	10	☑	C3	0%	0%	**C3**	0%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	C3
6	C4	50	10	☑	C4	0%	0%	0%	**C4**	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%	C4
7	N01	30	3	☐	N01	1%	1%	1%	1%	**N01**	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	N01
8	N02	30	3	☐	N02	1%	1%	1%	1%	1%	**N02**	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	N02
9	N03	30	3	☐	N03	1%	1%	1%	1%	1%	1%	**N03**	6%	6%	6%	6%	6%	6%	6%	6%	6%	N03
10	N04	30	3	☐	N04	1%	1%	1%	1%	1%	1%	1%	**N04**	6%	6%	6%	6%	6%	6%	6%	6%	N04
11	N05	30	3	☐	N05	1%	1%	1%	1%	1%	1%	1%	1%	**N05**	6%	6%	6%	6%	6%	6%	6%	N05
12	N06	30	3	☐	N06	1%	1%	1%	1%	1%	1%	1%	1%	1%	**N06**	6%	6%	6%	6%	6%	6%	N06
13	N07	30	3	☐	N07	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	**N07**	6%	6%	6%	6%	6%	N07
14	N08	30	3	☐	N08	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	**N08**	6%	6%	6%	6%	N08
15	N09	30	3	☐	N09	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	**N09**	6%	6%	6%	N09
16	N10	30	3	☐	N10	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	**N10**	6%	6%	N10
17	N11	30	3	☐	N11	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	**N11**	6%	N11
18	N12	30	3	☐	N12	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	**N12**	N12

Tablo 4: Örnek sahne çalışması yönetici ekranı.

	A	B	C	D	E	F	G	H	I	J	K
1	**Süre (saniye)**	**30**						**Bağlantı Modifikasyonları**			
2	**Node**	**T**	**X**	**Y**	**Link K.**	**Kütle**		**1. Node**	**2. Node**	**Bağlı İken**	**Bağlı Değilken**
3	C1		20%	70%	2	100		C1	N01		6%
4	C2		70%	20%				C2	N02	20%	
5	N01		30%	30%	2						
6	N02	10									
7											
8											
9											



Şekil 47. Ön prototip uygulamanın örnek sahne çalışması anından ekran görüntüsü.

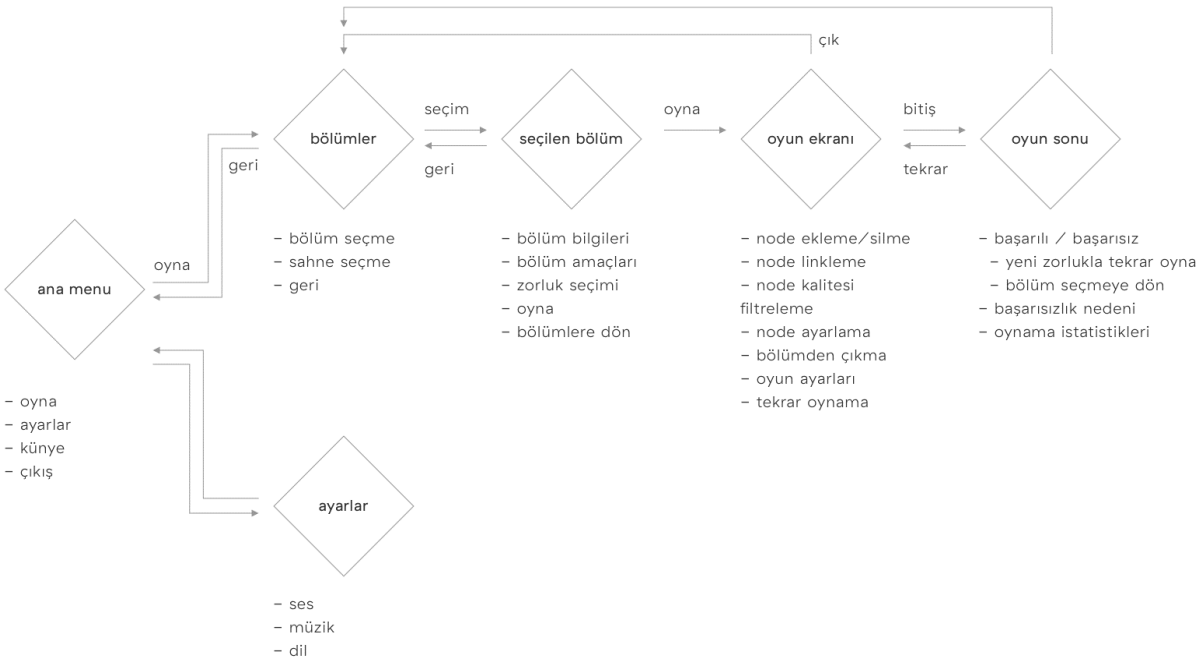
Prototip uygulama aşamasının ardından kütle çekim yasası ile elemanlar arasındaki anlık kuvvetleri hesaplayan bir yöntemi temel alarak geliştirilen uygulamaya grafik tasarım elemanları dahil edilmiş, arka planda Mekan Dizimi hesaplamalarını yapabilir hale getirilmiş, oluşan ağ yapılarının SpaceChase for Grasshopper ile birlikte işlenebilmesi sağlanmış ve oyuncuların oyun deneyimlerinin anonim olarak kaydedilmesi mümkün kılınmıştır. Bu gelişmelerle birlikte sonuç halini alan SpaceChase oyununun güncel sürümünün kullanım detaylarına devam eden bölümde yer verilmiştir.

4.2.3 Temel Kullanım Biçimi

Bu bölümde SpaceChase oyununun nasıl kullanıldığı, temel özellikleri ile özetlenmiştir. Anlatılan işlemlerin daha detaylı ve teknik açıklamaları *EK 15'teki SpaceChase Kullanım Kılavuzu*'nda sunulmuştur.

https://admin.spacechase.app/ web adresi aracılığıyla oyunun “yönetici (admin) arayüzü”ne erişilmektedir. Yalnızca oyun tasarım ekibi ve proje ekibinin erişimi olan bu adreste ilk olarak bir “giriş ekranı” ile karşılaşılır. Kullanıcı bilgileri ile giriş yapıldıktan sonra “Mekanlar/Spaces”, “Nodes/Düğümler”, “Düğüm Etkileşimleri/Node Influences”, “Uygulama/Application” ve “Oyuncular/Players” sekmelerinin olduğu bir ekrana yönlendirilir. Kullanıcı, mekanlar sayfasında “mekan yarat/create space” butonuna tıklayarak sahne oluşturur ve isimlendirir. Daha sonra oluşturduğu sahnelerin içine farklı hedeflerde ve zorluk seviyelerinde bölümler oluşturabilir ve her bir bölüm için o bölüme ait düğümleri, bölüm hedeflerini ve bölüm açıklamasını girebilir. “Düğümler” sayfasında ise “düğüm yarat/create node” butonu ile yeni düğümler yaratabilir, düğümlerin hangi mekana ait olacağını seçebilir, yaratılan düğümleri mekanlara göre filtreleyerek görüntüleyebilir. Düğüm ismi, ikonu, kalitesi, bağ sayısı, büyüklüğü gibi verileri de yine bu ekranda girebilir ya da gerektiğinde bu bilgileri yeniden düzenleyebilir. Aynı zamanda yöneticiler, “düğüm etkileşimleri” sayfasında ilgili sahneye ait “düğümler arası etkileşim matrisi” ile itme-çekme ilişkilerini girebilir. Yönetici arayüzündeki “Application/Uygulama” butonu ise kullanıcıyı oyun ekranına yönlendirmekte ve kullanıcıya oluşturduğu sahneleri/bölümleri hızlıca deneyimleme imkanı sunmaktadır. “Oyuncular/Players” sayfası ise oyunda kaydedilen oyuncu verilerine ve her bir oyuncunun oyun sonunda oluşturduğu ağ yapısı üzerinden ilgili Mekan Dizimi metriklerini edinmeyi sağlayan erişilebilirlik grafiklerine ulaşmayı sağlamaktadır.

Oyuncu, https://play.spacechase.app/ bağlantısı aracılığıyla ***Oyun Arayüzü’***ne ulaştığında “ana menü” ekranı ile karşılaşmaktadır. Buradan oyuncu “play/oyna” butonuna bastığında sahnelerin/bölümlerin görüntülenebildiği “bölümler” ekranına yönlenmektedir. Burada oyuncu oynamak istediği sahne ve bölümü seçer. Bölümü seçtiğinde “seçilen bölüm” ekranına gelir ve burada o bölüm ile ilgili açıklamaya, bölümün amaçlarına ve zorluk seçimine ulaşır. Oyuncu seçimini yaparak “play/oyna” butonuna bastığında o bölümün “oyun ekranı”na gider. Oyun ekranında ilgili düğümler ve bağlamsal düğümler oluşmaya başlar, oyuncu bu düğümlerin üzerine tıklayıp imlecini hareket ettirerek noktalar (mekansal birimler) arasındaki ilişkileri kurar. Oynadığı bölümün oyun başlangıcında belirtilen hedeflerini tanımlanan süre içinde gerçekleyen oyuncu başarılı olur ve “bölümler ekran”ına geri döner, bu ekrandan sıradaki bölümü veya daha önce oynadığı bir bölümü farklı bir zorlukta oynamayı seçebilir (Şekil 48).



Şekil 48. SpaceChase ekran akış diyagramı.

Uygulama ekranlarının yanı sıra ***Oyunun Temel Kuralları*** da uygulamanın kullanım biçimini şekillendirmektedir. Bu kurallar şu şekilde özetlenebilir:

A. Oyun ekranı iki alandan oluşmaktadır; oyun alanı ve dış alanı. Düğümler sadece oyun alanında bulunabilmekte, bağlamsal düğüm noktaları ise hem oyun alanında hem de dış alanda olabilmektedirler. Bağlar ise her iki alanda da bulunabilmektedir.

B. Düğümler, itme-çekme kuvvetlerinden ötürü oyun alanından çıkabilirler. Oyun alanından çıkan düğüm yok olmakta ve oyuncuyu başarısızlığa götürmektedir.

C. Düğümler devamlı hareket halinde olduğundan ancak bir bağlamsal düğüm noktasına bağlanırsa oyun alanından çıkması engellenmektedir. Bu tür düğümler kalıcı *[prominent]* olmaktadırlar. Kalıcı olan bir düğüme bağlanan kalıcı olmayan *[non prominent]* bir düğüm kalıcı hale gelmektedir.

D. Oyunda bazı düğümler senaryoya bağlı olarak daha “ağır”dır (örneğin bir hastane için “acil”in kolay değişmeyen kalıcı ve başat bir program olması gibi) ve bu düğümler itilme çekilme kuvvetlerinden daha az etkilenmektedir. Ancak bu tür düğümler kalıcı olmadıklarında yine de oyundan çıkma ihtimaline sahip olmaktadır (hastanenin bir estetik klinik olması ve acilin önemsizleşmesi gibi).

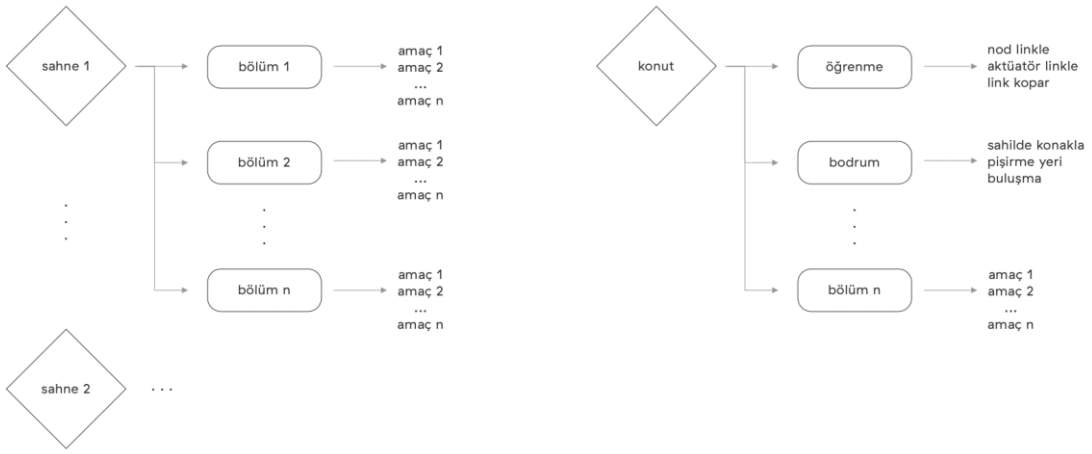
E. Oyuncu, oyun alanı içinde düğümlerin yerlerini tutup çekerek değiştirebilmektedir. Kalıcı olmayan düğümler oyun dışına taşınırsa yok olmaktadır.

F. Oyuna başlangıç anı t=0 olarak referans alınmaktadır; bu anda oyun alanında hazır düğümler ve bağlar bulunabileceği gibi oyun boş bir oyun alanıyla da başlayabilmektedir. Bu durum her bölümün nasıl kurgulandığına bağlı olarak değişmektedir.

G. Oyuncu, (oynanan bölüm el verirse) oyun alanına yeni düğümler ekleyebilmektedir. Hangi düğümlerin kaç adet eklenebileceği bölümün nasıl kurgulandığına (bölüm tasarımına) göre değişiklik göstermektedir.

H. Oyuncu hamlesiyle düğümleri yok edebilmektedir. Bağlamsal düğümler ise oyuncu tarafından yok edilememektedir.

Oyunun ***Bölüm Yapısı***'na bakıldığında; oyunun sahnelerden, sahnelerin ise farklı zorluk seviyelerine sahip bölümlerden oluştuğu görülmektedir. Oyunun bugüne kadar geliştirilen prototiplerinde 3 sahne (konut, restoran ve kreş) üzerine çalışılmıştır (Şekil 49). Her sahnenin ve bölümlerin hikayeleri ve kullanılan düğüm ve bağlamsal düğüm tiplerine göre birbirlerinden ayrılmaktadırlar. Örneğin "konut" sahnesinde 'uyuma', 'pişirme', 'yoga' gibi düğümler olabilirken, "restoran" sahnesinde 'karşılama', 'buzluk', 'üst değiştirme' gibi düğümler olabilmektedir.



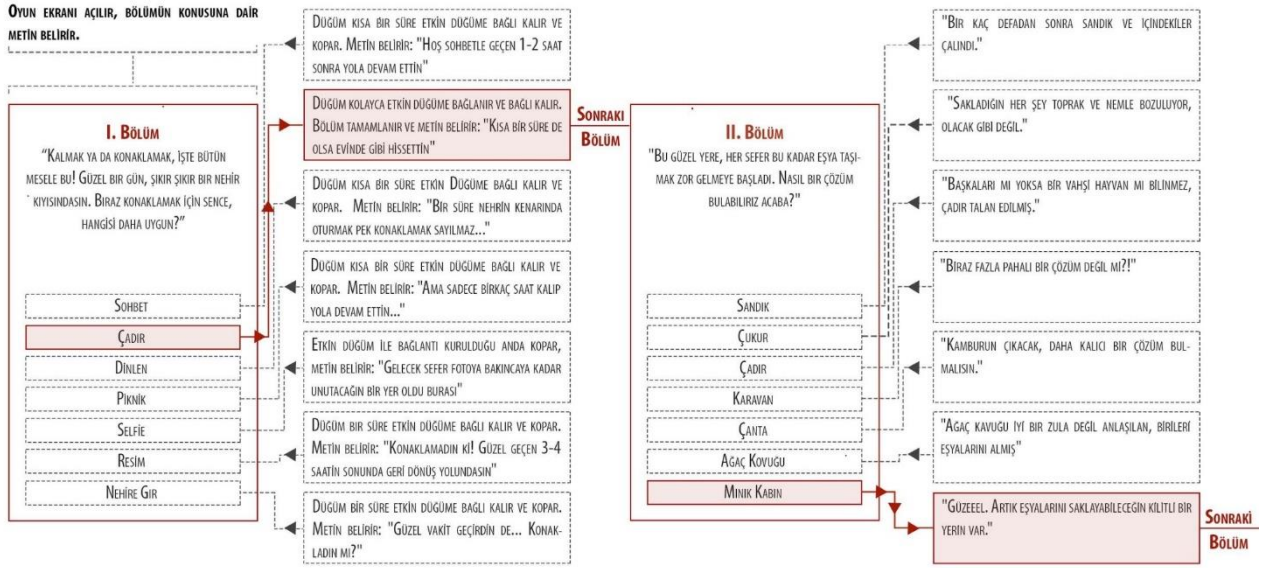
Şekil 49. Oyunun bölüm yapısı akış diyagramı.

Her sahne bölümlerden oluşmaktadır. Oyuncu bölümleri doğrusal bir sırayla oynar ve bölüm ilerledikçe bölümlerin karmaşıklığı ve zorluk seviyesi artar. Ayrıca her bir bölüm de normal, zor, çok zor olmak üzere 3 zorluk seviyesinden oluşur. Oyuncu bir bölümü en az normal zorlukta bitirirse sıradaki bölüme geçebilmektedir.

Bölümler ve bölüm zorlukları nasıl tasarlandıklarına göre değişiklik gösterir. Bölümler tasarlanırken oyun yöneticisi (admin) aşağıdaki ayarların/yöntemlerin birini veya birkaçını kullanabilir. Bu ayarların oyunun arayüzünde karşılıkları bulunmaktadır.

- Düğüm/bağlamsal düğüm kalitelerinin bir bağlanmaya veya zamana göre değişmesi.
- Oyuncuya tüm veya bazı düğüm kalitelerini değiştirme yetkisinin verilmesi.
- Oyun alanında belli zamanlarda yeni düğümlerin belirmesi.
- Zaman limiti: Oyuncuya verilen bölüm amaçlarının, verilen zaman limitinde gerçekleştirilme gerekliliği.
- Oyuncunun belirli sayıda veya sonsuz sayıda yeni düğüm/bağlamsal düğüm ekleyebilmesi.
- Bölüm durumları: Oyuncunun kararlarını etkileyebilecek atmosferik koşullar veya geneli etkileyen temalar, hava durumu, mevsim, ya da pandemi gibi, belirlenmesi. Varsayılan şekilde bölümler tek duruma sahiptir. Birden fazla duruma sahip bölümlerde her durum için belli düğüm/ bağlamsal düğümlerin yerleri ve kaliteleri değişebilir. Örneğin, bir bölümde ‘yağmurlu’ ve ‘güneşli’ olarak 2 durum olabilir ve bunların seçimine göre bazı bağlamsal düğümler yer değiştirebilir.

Oyun yukarıda bahsedildiği gibi üç sahne ve her sahnenin sahip olduğu en az üç bölüm üzerinden tasarlanmıştır. ***Sahne ve Bölüm Metinleri,*** kapsamında oyun tasarımı kurgusu gereği oyunu meydana getiren sahne ve bölümleri oluşturacak olası hikayeler için profesyonel metin yazarlığı hizmeti alınmıştır. Oluşturulan hikayelerin, özgün bir kurguda, dikkat çekici, eğlenceli ve kimi durumlarda oyunu meydana getiren sıralı sahnelerin devamı niteliğinde olması amaçlanmıştır (Şekil 50). Metin yazarlığı ile hedeflenen bir diğer önemli nokta ise mimarlıkta ağ düşüncesi ve Mekan Dizimi teorisine temellenen böylelikle bilimsel bir arka planı olması hedeflenen oyunun teknik terimlerinin, oyuncu ile güçlü bir iletişim kurmak üzere gündelik dile nasıl tercüme edilebileceğinin araştırılmasıdır. Oyuncunun bu terimlere dilerse ulaşabildiği bir raporlama, blog ya da web sitesi iletişiminin olması ve ayrıca Gelişmiş Mod (Advanced) oyun arayüzü kurgulanması gelecek çalışmalar için hedeflenmektedir.



Şekil 50. Oyuncunun tercihleri ile süren örnek sahne ve bölüm akış diyagramı.

4.2.4 Sınırlar ve Olanaklar

Oyunda hedeflenen oyuncunun etkileşimindeki çözük ağ matrisinin bir yanda Mekan Dizimi analizlerine bağlı tepki üretebilmesi ve hikayelerinin de mimari regülasyonlara ya da mevcut bina şemalarına atıfta bulunarak düzenlenebilmesidir.

- **Sınırlar**

- Oyunun hitap ettiği grupların geniş tutulması, dilinin mimari jargonlardan uzak ve kolay anlaşılır olmasını gerektirmiştir.
- Yönetici ekranının çok katmanlı olması kullanışlılığını azaltmakta ve her iki ekran arasında sıkça gidip gelme gerekliliği oyun tasarımı için zamana ve oyun geliştirme ekibinden yoğun beta testlere ihtiyaç doğurmaktadır.
- Oyunun web üzerinden işliyor olması, internet kalitesine bağlı olarak yavaşlamalara sebep olabilmektedir.
- Oyun seanslarının kaydı henüz çok yer kaplamamakta ancak gelecekte hem dosya alanı hem de ilgili araştırmacı gereksinimine işaret etmektedir.
- Mevcut durumunda oyun, Mekan Dizimi'nin gerçek zamanlı ölçümlerini yaparak dinamik grafa yön verebilir durumda değildir. Ancak bunun için gerekli alt yapı hazırdır.

- **Olanaklar**

- Düğümlerin ve bağların görsel dilinin inceltilmesi, başta tasarlanan etkileşim olanaklarının geliştirilmesi için gerekli prototipleme hazırdır. Gelecekte hızla yeni sahneler tasarlanarak etkileşim araştırması yapılabilir durumdadır.
- Oyun seanslarının verilerinin kaydedilmesi kullanıcılar arasında kıyaslamalı bir analiz olanağı verebilmektedir.
- Mekan Dizimi analizlerinin gerçek zamanlı olarak oyuna aktarımı için gerekli alt yapı hazırdır. Gelecekte bu kalite ile mekansal ağın hareketi strüktüre edilebilir.
- Mekan Dizimi analizleri ile depolanan kullanıcı verisi oyunun bilimsel bir araştırmaya altlık oluşturması anlamında değerlidir.

4.3 Yazılım Testleri ve Geri Bildirimler

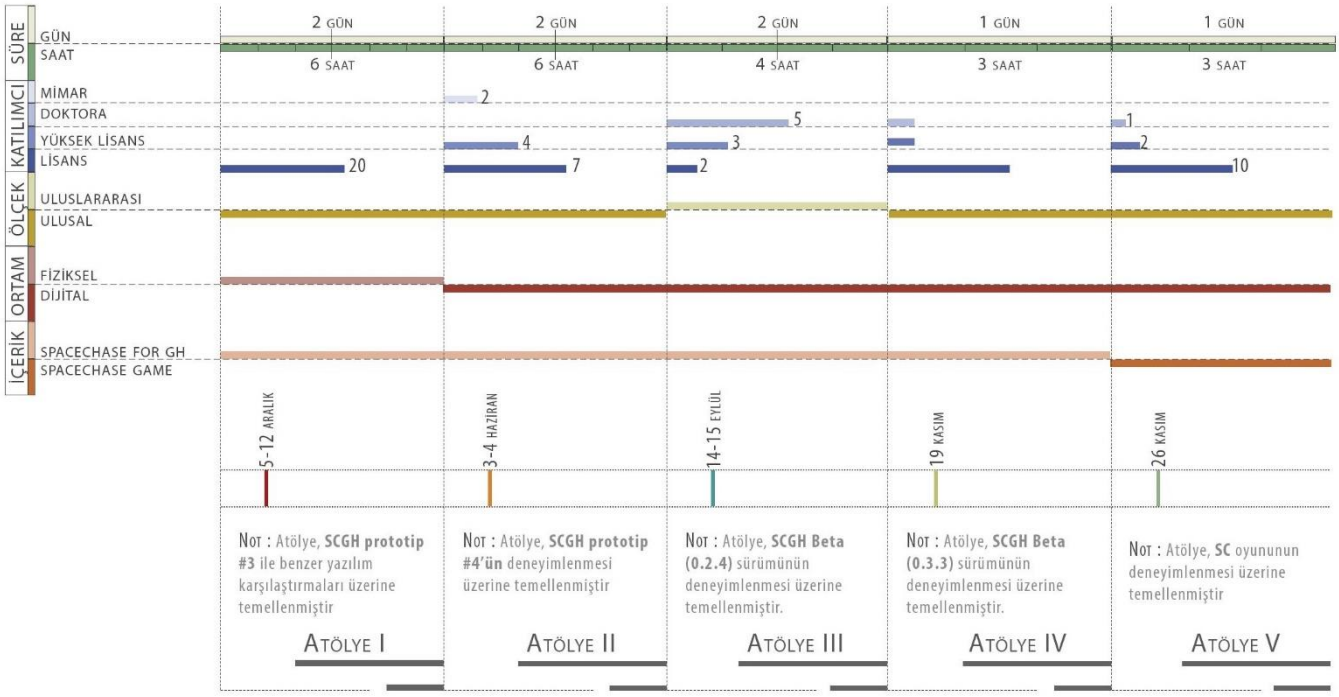
Bu bölüm, proje süresince ağırlıklı olarak mimarlık öğrencilerinin katılımıyla gerçekleştirilen atölyeleri, atölyelere ilişkin değerlendirmeleri, proje ekibi içinde yürütülen yazılım testlerini, ulusal ve uluslararası ölçekte konunun uzmanı akademisyenler ve profesyoneller ile tartışıldığı bilimsel toplantıları ve projeyi tanıtmak üzere kurgulanan web sitesi ve sosyal medya kullanımına ilişkin çalışmaları kapsamaktadır.

4.3.1 Atölyeler ve Yazılım Testleri

Proje kapsamında geliştirilen *SpaceChase for Grasshopper* eklentisi için 5-12 Aralık 2019, 3-4 Haziran 2020, 15-16 Eylül 2020, 19 Kasım 2020 tarihlerinde; *SpaceChase* oyunu için ise 26 Kasım 2020 tarihinde, ulusal ve uluslararası ölçekte toplamda beş farklı atölye çalışması gerçekleştirilmiştir (Şekil 51). Geliştirilen eklenti ve oyunsal yazılım, mimarlık ve tasarım pratiğinden farklı düzeyde katılımcıların dahil olmasıyla düzenlenen bu atölye serisinde deneyimlenmiştir. Katılımcılardan proje kapsamında geliştirilen yazılımların farklı versiyonlarına ilişkin geri dönüş almak, katılımcı görüşleri üzerinden gözlem yapmak ve ilerleyen süreçte uygulamanın olası test ekibini belirlemek atölye serisinin temel amaçlarını oluşturmaktadır. Bu amaçlar doğrultusunda atölye serisi süresince, atölyelerde yazılımların temel çalışma prensipleri ve oyunsallaştırılma kapasiteleri üzerinde durulmuş, mimarlıkta ağ düşüncesine odaklanılarak geliştirilen grafik-teorik araçlarla mekansal biçimlenmelerin ürettiği potansiyeller deşifre edilmeye çalışılmıştır.

Atölyeler süresince katılımcılar ve görüşleri, gözlem, mülakat, anket ve video-ses kaydı teknikleri ile kaydedilmiş ve bu kayıtlar için katılımcıların “bilgilendirilmiş onam formu” aracılığıyla

izinleri alınmıştır. Atölyeler sonucunda öğrencilerin yazılımların işleyişine, potansiyellerine, kısıtlarına ilişkin değerlendirmeleri derlenmiş ve proje sürecinde değerlendirilmiştir.



Şekil 51. Atölyelere ilişkin veri analiz şeması.

Yaklaşık bir yıllık bir süreye yayılan atölyelerin programları ve işleyişleri, zaman içinde projenin ve yazılımların gelişmesine paralel olarak farklılaşmıştır. Genel hatlarıyla, atölyelerde araştırma projesinin kapsamı, Mekan Dizimi teorisi, araç setleri, metrikleri ve mimarlıkta ağ düşüncesi üzerine kısa anlatımlar gerçekleştirilmiştir. Ardından katılımcılara yazılım/ların tanıtımı yapılarak işleyiş biçimi aktarılmış ve verilen tasarım problemleri bağlamında yazılım/ları deneyimlemeleri sağlanmıştır. Yapılandırmacı bir pedagojik perspektif ile yürütülen tasarım stüdyolarında uygulanan kısa süreli ve bağlamı kısıtlanmış tasarım egzersizlerinde olduğu gibi, bu atölye çalışmalarında da katılımcıların belirli kısıtlar içinde kendilerine verilen tasarım problemine çözüm aramaları beklenmiştir. Atölyelerde katılımcılara verilen problemler, kapsamlı ve karmaşık mekan tasarımları değil, mekanlar arasındaki ilişkileri temel düzeyde sorgulamayı ve sebep-sonuç ilişkilerini gözlemlemeyi hedefleyen egzersizler olarak kurgulanmıştır. Bu süreçte atölye kurgularının kendisi de, yazılım/ların oyunsallaştırılma kapasitesinin bir bileşeni olarak değerlendirilmiştir.

Proje süresince geliştirilmekte olan SpaceChase for Grasshopper ilk kez birinci atölyede *prototip #3* versiyonu ile paylaşılmıştır. *Atölye I*, geliştirilen yazılımın yanı sıra mimaride ağ düşüncesini tartışmaya açan *ikide1* oyunu ve mevcut benzer iki yazılımın (*Cytoscape ve Syntactic*)

irdelenmesini de kapsayacak şekilde kurgulanmıştır. (EK 16) İkide1 oyunu, belirli bir kural setine ve mekansal ilişkilere odaklanan dinamik bir sistem oluşturarak ağ düşüncesinin katılımcılar tarafından bedensel olarak deneyimlenmesine olanak sağlamıştır (Şekil 52). Cytoscape ve Syntactic yazılımları ise farklı ağ ölçüm metriklerine ve görselleştirme tekniklerine sahip programlar üzerinden konuyu etraflıca değerlendirmeyi mümkün kılmıştır. Son olarak SpaceChase for Grasshopper'ı bir tasarım sorusu eşliğinde deneyimleyen katılımcılar, geliştirilen yazılımı karşılaştırmalı olarak değerlendirmişlerdir (Şekil 53).



Şekil 52. İkide1 Oyunu: oyun anından görüntüler.



Şekil 53. Atölye I'de incelenen üç programa ilişkin karşılaştırmalı katılımcı değerlendirmeleri.

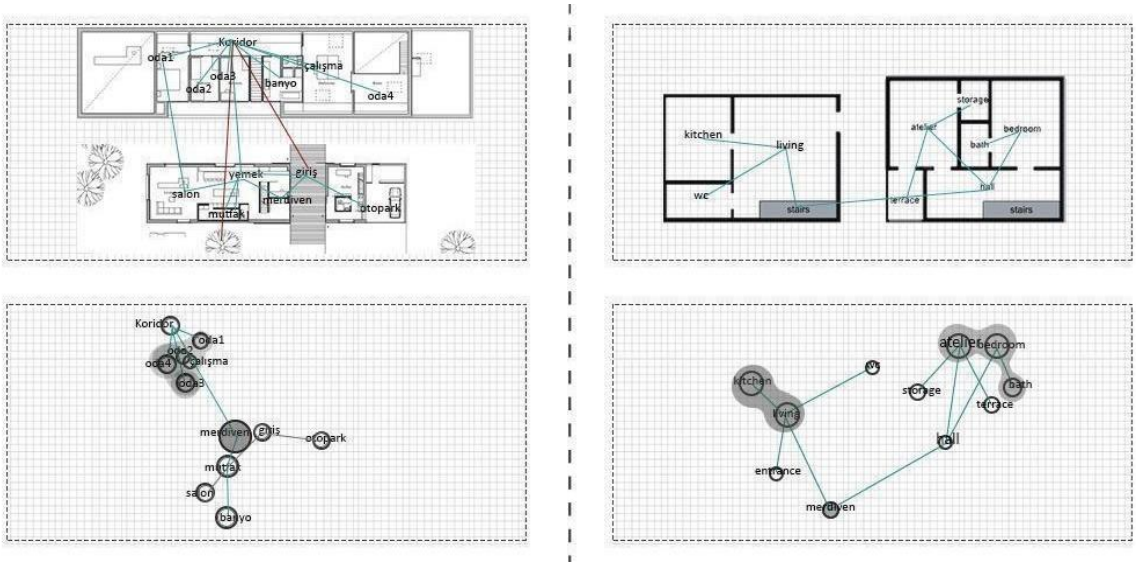
SpaceChase for Grasshopper'a ilişkin öğrencilerin öne çıkan yorumları ise aşağıdaki gibi sıralanabilir;

- Öğrencilerin yorumlarında SpaceChase for Grasshopper için en çok kullanılan anahtar kelimeler "dinamik", "ilginç" ve "etkili" olarak belirlenmiştir.
- Öğrencilerin %50'si, mekansal ilişkilerin bütünlüğünü görebilmek açısından yazılımı etkili bulmuştur.

- Katılımcıların yaklaşık %70'i ileride mimari tasarım süreçlerinde bu uygulamayı kullanmak istediklerini, yaklaşık %60'ı ise bu yazılımı mimarlık öğrencisi arkadaşlarına önermek istediklerini belirtmiştir.
- Öğrencilerin %30'unun ortak eleştirisi, yazılımın Grasshopper ve Rhinoceros programlarının kullanım bilgisini gerektirmesi olduğu yönündedir.

Öğrencilerin, uygulamayı kullanmanın zorluğunu ve ihtiyaç duydukları bilgilerin erişilebilirliğini eleştirdikleri görülse de bu uygulama ile geçirdikleri zamandan büyük ölçüde memnun kaldıkları gözlemlenmiştir.

Detaylı programına EK 17 üzerinden ulaşılabilen *Atölye II* kapsamında katılımcılar, birer mimar ve oyuncu olarak SpaceChase for Grasshopper *prototip #4*'ü deneyimlemişlerdir. Öncelikle verilen örnek bir konut şeması üzerinden modelleme yaparak yazılımın kullanım aklını kavrayan katılımcılar, devamında kendi konut tasarımlarını/sahnelerini yine aynı yazılım aracılığıyla üretmişlerdir (Şekil 54). Kullanım sürecinde karşılaşılan olumlu ve olumsuz yönleri tespit etmek ve katılımcıların birbirlerinden öğrenmesini desteklemek adına katılımcılardan yazılımı kullanım süreçlerini ve üretimlerini sunmaları istenmiştir.



Şekil 54. SpaceChase for Grasshopper ile katılımcılar tarafından üretilen sahne örnekleri.

Atölye sonunda sürece yönelik yapılan görüşmeler doğrultusunda önce çıkan yorumlar şöyle özetlenebilir;

- Katılımcı A, yazılımın yapım aşamasında olduğu için kullanım zorlukları barındırdığını ve çalışmanın geliştirilmesi sonucunda yazılımın kullanıcının mimari tasarım sürecinde algı potansiyellerini yükselten bir tasarım aracı olarak kullanılabileceğini belirtmiştir.

- Katılımcı B, geliştirilen yazılımın mimari tasarım sürecinde kullanıcı için fikir opsiyonlarını çoğaltarak tasarıma ilişkin karar verme sürecine katkı sağladığını belirtmiştir. Ayrıca yazılım aracılığıyla kurulan mekan ilişkilerinin nicel veriler üretmesini, tasarımı analitik sonuçlara ulaştırması açısından olumlu bulmuştur.
- Öğrenci değerlendirmelerinde eklenti için en çok kullanılan anahtar kelimeler "esnek", "kullanışlı" ve "pratik" olarak belirlenmiştir.
- Yapılan anket çalışmasına göre, katılımcıların yaklaşık %25'inin verilen görevlerde kendilerini çok başarılı yaklaşık %60'ının ise başarılı olarak gördüğü tespit edilmiştir. Katılımcıların yaklaşık %85'i programın mesleki hayatlarında çok faydalı olacağını belirtmiş ve yaklaşık %40'ı verilen görevlerde kullandıkları programın çok faydalı olacağını ifade etmiştir.

Diğer atölye çalışmalarından farklı olarak *Education and Research in Computer-aided Architectural Design in Europe* (eCAADe) 2020 konferansı kapsamında uluslararası bir ortamda gerçekleştirilen *Atölye III* ile katılımcılar SpaceChase for Grasshopper Beta (0.2.5)'yı deneyimlemişlerdir. EK 18'den detaylı programına ulaşılabilen atölye III, Atölye II ile benzer olarak kurgulanmıştır. Aynı zamanda yazılımın önceki versiyonları ile gerçekleştirilen testlerin sonuçları, aynı konferansta bir bildiri olarak sunulmuş ve bu etkinlik kapsamında atölye sonucunda ve bildiri sunumunda projeye ilişkin geri dönüşler alınmıştır. Bu atölye kapsamında öne çıkan yorumlar aşağıdaki gibi özetlenebilir;

- Katılımcı A, tasarlama sürecinde eklentinin tasarlayan özne ile birlikte düşünen bir araç görevi gördüğünü vurgulamış ve tasarımcının meydana getirdiği konfigürasyonel olasılıklar arasında uygulamanın sahip olduğu bilimsel veriler aracılığıyla bir karşılaştırma fırsatı elde ettiğini belirtmiştir.
- Katılımcı B, kurulan mekansal organizasyonların analiz seti ile incelenerek tasarımcıya anlık geri dönüş vermesinin olumlu olduğunu dile getirerek uygulamanın tasarım sürecine dahil olan bir araç görevi gördüğünü vurgulamıştır. Uygulamanın, kullanıcıya aktardığı ifade tekniğinin diagramatik olduğunu ve SpaceChase for Grasshopper'ın kurgulanan mekan sahnesinde düşey ilişki kuramadığını dile getirmiştir. Bu nedenle karmaşık mekan sahnelerinde, uygulama kullanımının zorlaşacağı yönünde öngörüsünü paylaşmıştır.

Atölye IV kapsamında ise katılımcılar tarafından SpaceChase for Grasshopper Beta (0.3.3) deneyimlenmiştir. Bu atölyede öncekilerden farklı olarak katılımcıların konut şeması ile değil, gerçek bir tasarım problemi ile uğraşmaları hedeflenmiştir ve Bodrum Kızılağaç mevkiinde tasarlanması planlanan spor tesisi üzerinden yazılımı deneyimlemeleri beklenmiştir. (EK 19) Daha kapsamlı bir ihtiyaç programı sunan spor tesisini çalışmak üzere şu adımlar takip edilmiştir; (1) spor tesisi şemasında bağlam oluşturma potansiyeli olan çevresel verilerin

katılımcılara anlatılması, (2) spor tesisinin ihtiyaç programının katılımcılara verilerek kurulacak ilişki şemasının kısıtları ve gereklilikleri aktarılması, (3) çevresel veri olarak girilen bağlamsal düğüm noktalarının itme-çekme ilişkilerini belirleyebilmek adına katılımcılara spor tesisini deneyimleyecek üç farklı personanın tanımının yapılması, (4) öğrencilerin ürettikleri sahneler üzerinden mekansal ilişkilerin program yardımıyla görsellenerek analiz edilmesi, deneyimlenmesi istenir. Bu kapsamda katılımcılar, kendi mekan organizasyonlarını oluşturmuşlardır.

Katılımcıların yazılım aracılığıyla ürettikleri şemaları sunmalarının ardından ortak bir tartışma ortamında değerlendirmeleri ve görüşleri alınmıştır. Öne çıkan yorumlar aşağıdaki gibi özetlenebilir;

- Katılımcı A, SpaceChase for Grasshopper uygulaması aracılığıyla mekanlar arasında kurulan ilişkilerde birçok ihtimali bir arada görmenin olumlu olduğunu dile getirmiş, tasarlama sürecinde fark edilemeyen sezgisel kararların bu yazılım aracılığıyla deşifre edilerek görünür hale geldiğini aktarmıştır.
- Katılımcı B, SpaceChase for Grasshopper uygulamasının tasarım süreci için bir sonuç ürün oluşturmadığını, tasarımcıya olası potansiyelleri/imkanları görünür kıldığını belirtmiştir. Tasarlanan mekanın ve mekanda gerçekleştirilen deneyimin öznel olduğuna vurgu yapmıştır. SpaceChase for Grasshopper uygulamasının bu öznel durumu arka plana attığını, mekansal ilişkileri sistematize ederek ön plana çıkardığını aktarmıştır.

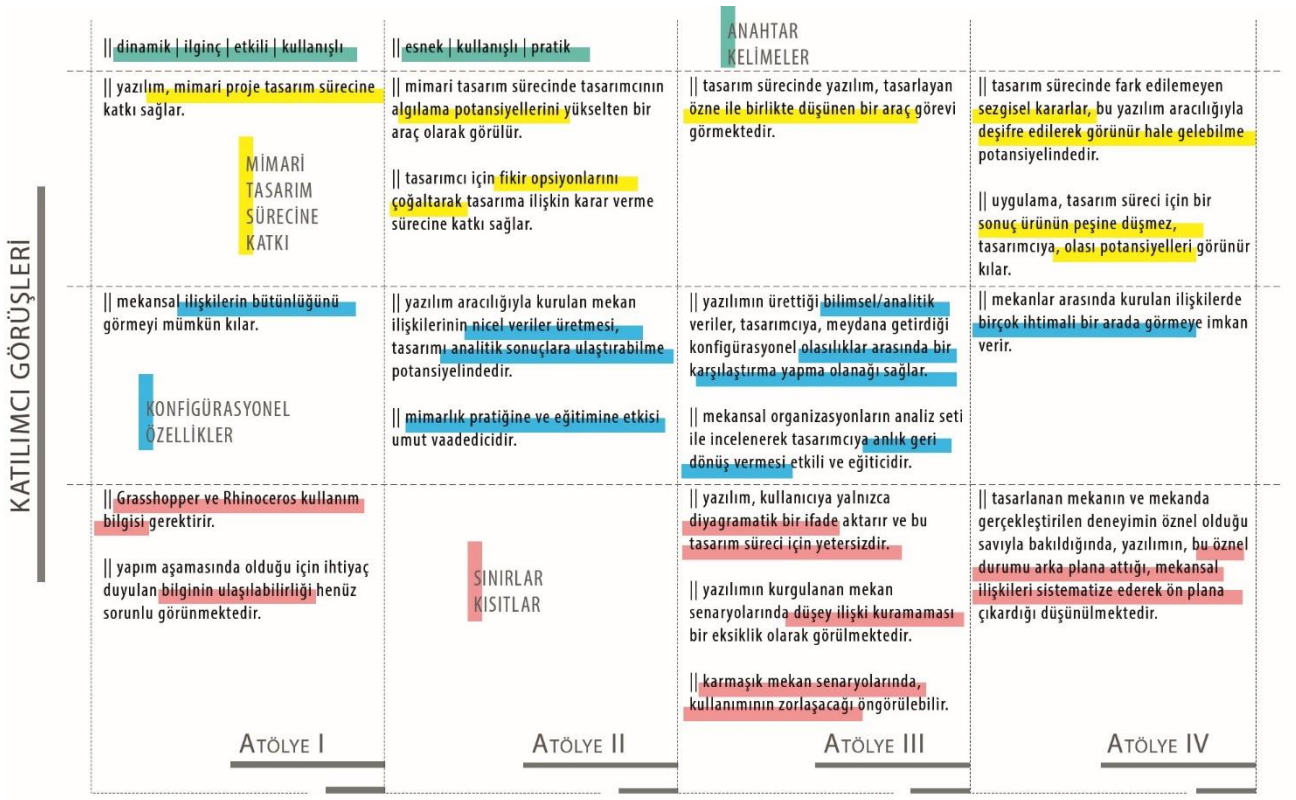
Atölye IV'ün devamı niteliğinde kurgulanan ve aynı katılımcılarla devam eden *Atölye V* ise proje kapsamında geliştirilen SpaceChase oyununun ilk kez paylaşıldığı bir ortam olmuştur. (EK 19) Katılımcılardan öncelikle yazılım aracılığıyla belirli bir ihtiyaç programı dahilinde bir mekan listesi ve bu mekanlar arasındaki ilişkileri gösterir nitelikte bir matris kurgulamaları istenmiştir. Katılımcıların hem bir mimar hem de oyuncu olarak, *SpaceChase* oyununda kurguladıkları ilişkilere ek olarak mekansal davranışları/kaliteleri sisteme katmaları ve mekansal ilişkileri dinamik olarak görselleştiren diyagrama müdahale etmeleri beklenmiştir. Eş zamanlı olarak üretilen ağlardaki mekansal kaliteler oyunun yönetici arayüzü aracılığıyla analiz edilerek öğrencilere görsel ve sayısal geri bildirimler verilmiştir. Bu aşamada toplanan veriler, öğrencilerin yazılımı kullanırken ürettiği mekansal ilişki ağlarını, bunların analiz sonuçlarını ve kullanıcıların bu sonuçlar üzerinden yaptıkları seçimleri içermektedir. Atölye sonucunda katılımcılardan sunumları aracılığıyla görüşleri alınmıştır. Öne çıkan yorumlar aşağıdaki gibidir;

- Katılımcı A, zaman faktörü nedeniyle bilinçli bir şekilde mekansal ilişki kurmanın zor olduğunu yönünde eleştiride bulunurken oyun sonunda sentaktik veriler ile kurduğu şemanın

analiz edilmesini kurguladığı mekansal birlikteliklerin kalitesini anlamasına olanak verdiği için olumlu bulmuştur.

- Katılımcı B, oyunun tanımlanmış sahneler üzerinden ilerlemesinden ziyade, oyuncu olarak kendi sahnesini kurma imkanına sahip olabilmeyi ve böylece oyunun bireysel tasarlama sürecinde de bir araç olarak kullanılma potansiyeli oluşturabilmesini gerekli ve kıymetli gördüğünü belirtmiştir.

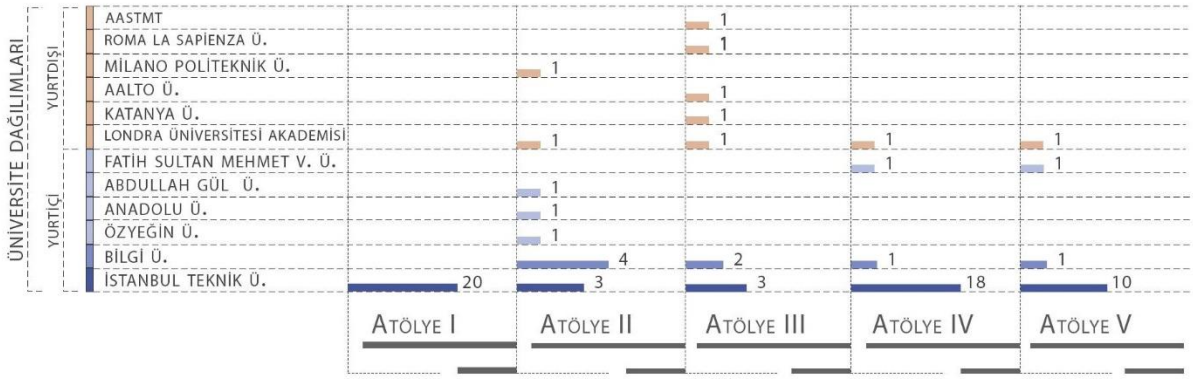
Sözü edilen atölyeler kapsamında katılımcı değerlendirmeleri çeşitli yorumlar ve anahtar kavramlar üzerinden şekillenmiş ve yazılım geliştirme çalışmalarına rehberlik etmiştir (Şekil 55). Bu yorumlar ana hatlarıyla geliştirilen araçların mimari tasarım sürecinde kullanılabilirliği, barındırdıkları potansiyelleri, mekanın konfigürasyonel özelliklerine ilişkin ne tür kaliteler ortaya koyabilecekleri ve de sahip oldukları sınırlar üzerine odaklanmıştır.



Şekil 55. Atölyelerde öne çıkan katılımcı değerlendirmelerinin dökümü.

Atölyeler, farklı donanımlara sahip mimarlık öğrencilerden ve mimar kişilerden yazılımlara ilişkin geri bildirimlerin alınabilmesi için etkin deneme/yapma/öğrenme ortamları/olanakları kurgulamıştır. Fiziksel ortamda başlayan atölye serisi yaşanan gelişmelerle sanal ortama taşınmıştır. Fiziksel birlikteliklerin sınırlandığı bu ortamlar birbirinden öğrenme sürecini olumsuz etkilemiş olmakla birlikte farklı mekanlardan/okullardan kişileri bir araya getirebilmesi, sürecin tümüyle kayıt edilip izlenebilmesi, arşivlenebilmesi açısından da potansiyeller sergilemiştir. Bu

durum atölyelerden birinin uluslararası ortamda kurgulanabilmesine de olanak vermiştir. Bu sayede katılımcıların profillerinin çeşitlendiği görülür (Şekil 56). Yazılımlar daha çok lisans öğrencileri ile deneyimlenebilmiştir. Bu durum öğrencileri kuramsal tartışmalar ve program bilgileri anlamında zorlasa da araştırma konusuna, tasarımı öğrenen kişiler açısından bakılabilmesine ve değerlendirilebilmesine olanak vermiştir. Lisansüstü seviyedeki katılımcılar ise daha çok araştırmaları Mekan Dizimi, parametrik tasarım, algoritmik düşünme üzerine odaklanan kişilerdir. Böylelikle özellikle yazılımın işleyiş mekaniği konusunda daha donanımlı kişilerden değerli geri bildirimler alınabilmiştir.



Şekil 56. Üniversitelere göre katılımcı profili.

Atölyelerle birlikte araştırma grubu içinde yapılan testler de yazılımların geliştirilebilmesi için önemli geri bildirimler üretmesi anlamında çok değerlidir. Her bir test çıktılarıyla yazılımların kalitesini artırmak üzere değerlendirilmiştir. Bu denemeler üzerinden yazılımlar uzman kişilerin katkısıyla güncellenmiş, araştırma sürecinin dinamik bir yapıya sahip olması hedeflenmiştir.

4.3.2 Bilimsel Toplantılar

16, 22 ve 26 Aralık 2020 tarihlerinde gerçekleştirilen bilimsel toplantı serisi iki farklı kurguda, dijital ortamda organize edilmiştir (Şekil 57). Kasım ayı içinde planlanan bu etkinlikler yaşanan gelişmeler ve katılımcıların yoğun programları nedeniyle bu tarihlere ötelenmiştir. Bu toplantıların ilki, Mekan Dizimi kuramı ve hesaplamalı üretken tasarım üzerine odaklanarak çeşitli yazılımlar geliştiren, bu çalışmaya da önemli referans oluşturmuş araştırmacılarla bir araya gelmek, bir tartışma ortamı üretmek üzere kurgulanmıştır. İkinci olarak SpaceChase for Grasshopper eklentisi ve SpaceChase oyunsal yazılımı, Mekan Dizimi ve/veya oyun tasarımı üzerine çalışan/çalışmış akademisyenlerle paylaşılmıştır. Atölye serisinin devamını oluşturan bu organizasyonlarda temel amaç, tartışma ortamı yaratarak alınan geri bildirimlerle yazılımların geliştirilmesini sağlamak, yazılımların tasarım aracı olarak sahip olduğu kısıtlar ve potansiyeller

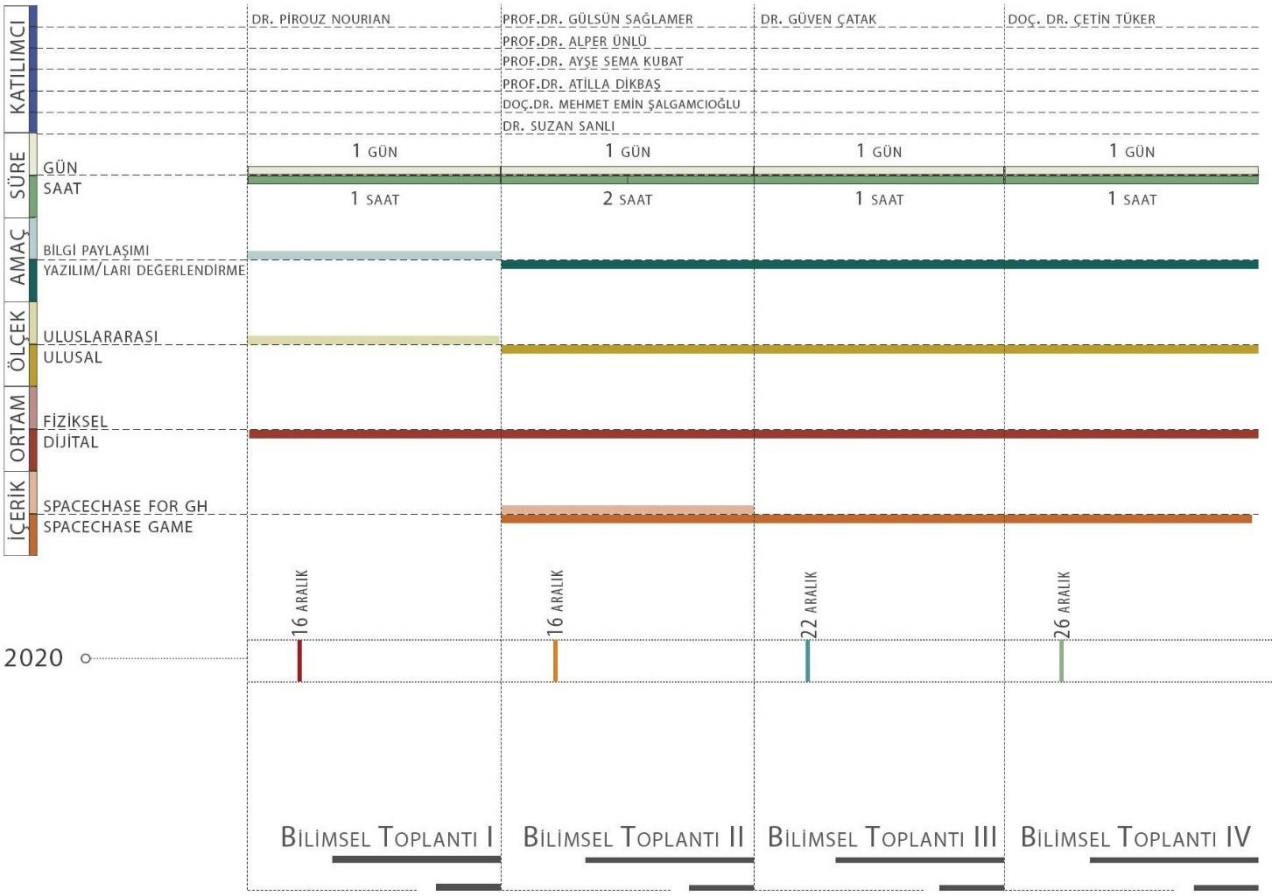
üzerine belirtilen görüşleri veri olarak kaydetmek ve gelecek çalışmalar için yol haritaları çıkarabilmektir.

Bilimsel Toplantı 1, TU Delft Üniversitesi'nden Dr. Pirouz Nourian'ın katılımıyla gerçekleşmiştir. Nourian, matematiksel modelleme, geometrik, topolojik veya grafik teorik hesaplama yoluyla hesaplamalı üretken tasarım konusunda çalışmalar yürütmektedir. Toplantı programı, Nourian'ın hesaplamalı üretken tasarım üzerine odaklanarak geliştirdiği çalışmalara dair sunuşu ile başlamıştır. Gerçekleştirilen sunuşta; "Diferansiyel denklemler aracılığıyla matematiksel olarak tasarım çözümleri nasıl üretilir?, Sentaktik yapılar aracılığıyla hesaplamalı olarak tasarım alternatifleri nasıl üretilir (keşfedilir / parçalara ayrılır)?, Katılımcı tasarım oyunlarında insan zekası ve açıklanabilir yapay zeka nasıl birleştirilir?" gibi sorular üzerinde durulmuştur. Nourian, yürütülen projede önemli bir referans noktası olan *Syntactic* yazılımının düşünsel ve teknik arka planını aktararak proje ekibi tarafından yöneltilen soruları cevaplamıştır. Diğer toplantılardan farklı olarak, uluslararası bir ölçekte Mekan Dizimi odaklı geliştirilen çalışmalar hakkında bilgi edinme fırsatı bulunmuştur.

Bilimsel Toplantı 2, bir akademisyen grubu ve uygulamadan bir mimarın katılımıyla (Prof.Dr. Gülsün Sağlamer, Prof. Dr. Atilla Dikbaş, Prof.Dr. Alper Ünlü, Prof.Dr. Ayşe Sema Kubat, Doç.Dr. Mehmet Emin Şalgamcıoğlu, Dr. Suzan Sanlı) 2 saatlik bir program dahilinde dijital ortamda gerçekleşmiştir. Prof.Dr. G. Sağlamer, Prof.Dr. A. Ünlü, Prof.Dr. A.S. Kubat, Doç.Dr. M.E. Şalgamcıoğlu Mekan Dizimi alanında çalışmalar gerçekleştirmiş, yüksek lisans ve doktora tezleri yürütmüş akademisyenlerdir. İTÜ Arı Teknokent Genel Müdürlüğü görevini yürütmekte olan Prof.Dr. A. Dikbaş, uygulamadan bir mimar olan ve doktora çalışmasını Mekan Dizimi alanında gerçekleştirmiş olan Dr. S. Sanlı katılımcılar arasındadır. Toplantı programı doğrultusunda, araştırma projesinin kapsamı, Mekan Dizimi teorisi ve mimarlıkta ağ düşüncesi üzerine bir bilgilendirme yapılarak proje kapsamında geliştirilen yazılımların düşünsel ve teknik arka planı karşılaştırmalı olarak katılımcılara aktarılmıştır. Bilgilendirme sürecinin tamamlanmasıyla birlikte yazılımların barındırdığı potansiyeller ve kısıtlar tartışmaya açılmıştır. Katılımcıların yorum ve kritikleri ile çalışmanın nasıl ilerleyebileceği sorgulanmıştır.

Bilimsel Toplantı 3, Bahçeşehir Üniversitesin'de Dijital Oyun Tasarımı Lisans Bölüm Başkanı ve Oyun Tasarımı Yüksek Lisans Program Koordinatörü olarak görev yapan Dr. Öğr. Üyesi Güven Çatak'ın katılımı ve *Bilimsel Toplantı 4*, Mimar Sinan Güzel Sanatlar Üniversitesi Grafik Tasarım Bölümü'nde öğretim görevlisi olarak görev yapan Doç. Dr. Çetin Tüker'in katılımı ile bir saatlik eş programlar dahilinde dijital ortamda gerçekleşmiştir. Bu toplantılarda, mimarlıkta ağ düşüncesi üzerine bilgilendirme yapılarak *SpaceChase* oyun yazılımın çalışma prensipleri ve

kuramsal arka planı aktarılmıştır. Toplantının devamında yazılımın sahip olduğu kısıt ve potansiyeller tartışmaya açılmış katılımcı görüşleri alınmıştır.



Şekil 57. Bilimsel toplantı serisi içerik şeması.

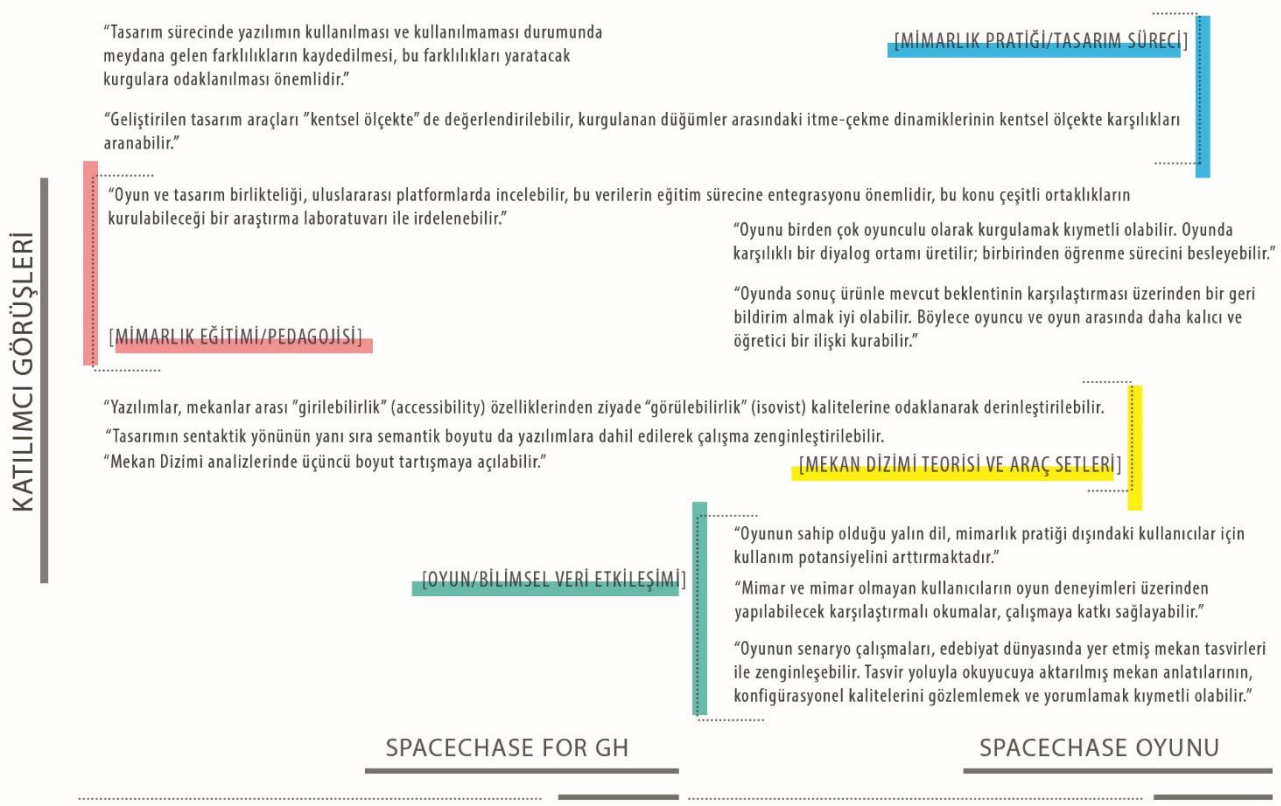
Bilimsel toplantı serisi boyunca kaydedilen önemli kritikler / değerlendirmeler şu şekildedir;

- Geliştirilen yazılımların kullanılması ve kullanılmaması durumunda tasarım sürecinde meydana gelen farklılıkların kaydedilmesi önemlidir. Bu nedenle ilerleyen süreçte gerçekleştirilecek atölye çalışmalarının bu farklılıkların tespit edilmesine yönelik kurgulanmasının proje için önemli bir veri oluşturacağı düşünülmektedir.
- Oyun ve tasarım birlikteliği üzerine uluslararası platformda yer alan öncül çalışmaların incelemesi, bu verilerin eğitim süreci içine nasıl entegre edilebileceği önemlidir; bu konu bir araştırma laboratuvarı kurularak irdelenebilir, çalışmalar bu laboratuvar ortamında çeşitli ortaklıklar kurularak yürütülebilir.
- Yapı ölçeğinde geliştirilen bu yazılımların kentsel ölçekte kullanılabilirliği de araştırmaya değerdir; araştırma bu noktada yeni bir perspektif kazanabilir. Geliştirilen tasarım araçlarında

kurgulanan düğümler arasındaki itme-çekme dinamiklerinin kentsel ölçekte karşılığının neler olabileceği sorgulanabilir.

- Geliştirilen yazılımların mekanlar arası girilebilirlik (accessibility) özelliklerine odaklandığı görülmektedir. Mekanın onu çevreleyen geometri ile ilişkisi ön plana alınarak araştırma, mekanlardan "görülebilirlik" (isovist) kalitelerine odaklanılarak derinleştirilebilir. Mekan Dizimi analizlerinde üçüncü boyut tartışmaya açılabilir. Geliştirilen yazılımlarda mekanı ören farklı katmanların sürece dahil edilmesi kıymetlidir; böylece tasarımın semantik boyutu derinleştirilerek araştırma projesi zenginleşebilir.
- Geliştirilen yazılımların bilimsel açıdan basit ve kullanılabilir olma özelliklerinin ilerleyen süreçte yitirilmemesi gerekir. Oyunun sahip olduğu yalın dil, mimarlık pratiği dışındaki kullanıcılar için kullanım potansiyelini arttırmaktadır. Bu doğrultuda mimar olan ve olmayan kullanıcıların oyun deneyimleri üzerinden yapılabilecek karşılaştırmalı okuma, araştırma sürecine katkı sağlayabilir.
- SpaceChase oyununun sahne çalışmaları zenginleştirilebilir; edebiyat dünyasında yer etmiş mekan betimlemeleri oyuna uyarlanabilir. Betimleme yoluyla okuyucuya aktarılmış mekan anlatılarının, konfigürasyonel kalitelerini gözlemlemek ve oluşturulan ağ benzeri yapıların nicel veriler ile incelendiğinde ne gibi potansiyellere gebe olduğunu yorumlamak kıymetli olabilir.
- SpaceChase oyunu tek bir oyuncu yerine, birden çok oyuncunun dahil olduğu bir süreçte oynanabilir. Bu özellik oyunda karşılıklı bir diyalog ortamı üretir; birbirinden öğrenme sürecini besleyebilir. Böylece oyun bir tasarım aracı olarak işlevselleşebilir.
- SpaceChase oyunu, oyuncunun oyun sırasında geri bildirim alması yerine, oyun tamamlandıktan sonra sonuç ürünle mevcut beklentinin karşılaştırması üzerinden bir geri bildirim alması şeklinde yeniden strüktüre edilebilir. Böylece oyuncu ve oyun arasında daha kalıcı bir ilişki kurabilir.
- Hayata geçmiş yapıların tasarım parametreleri oyun sahnelerine entegre edilerek oyuncu deneyimine sunulabilir ve sonuç ürünle mevcut durum karşılaştırılarak oyuncuya geri bildirim sağlanabilir.

Bilimsel toplantılar kapsamındaki değerlendirmeler çeşitli anahtar kavramlar üzerinden şekillenmiştir (Şekil 58). Genel hatlarıyla mimarlık pratiği ve mimari tasarım süreci, mimarlık eğitimi, Mekan Dizimi teorisi ve araç setleri, oyun ve bilimsel verinin etkileşimi konuları yürütülen araştırma ve geliştirilen yazılımlar dahilinde tartışılmıştır. Bu değerlendirmeler yazılım geliştirme çalışmalarına rehberlik ederken aynı zamanda gelecek çalışmalar için de yol gösterici olmuştur.



Şekil 58. Bilimsel toplantılar kapsamında öne çıkan değerlendirmeler.

4.3.3 Web Sitesi Üzerine Değerlendirmeler

Proje süresince yürütülen akademik çalışmaya ilişkin bilgi paylaşımın yapılması, geliştirilen yazılımların tanıtılması, gerçekleştirilen atölyelerin duyurulması, aynı zamanda bu etkinliklerin sonucunda üretilen bilgilerin paylaşılması ve akademik toplantılar için konuşmacılar, araştırmacılar ve katılımcılar ile iletişime geçilmesi amacıyla proje kapsamında bir web sitesi kurgulanmıştır. Web sitesi üzerinden ilerleyen bilgi paylaşım ve tanıtım çalışmaları çeşitli sosyal medya platformları aracılığıyla yapılan duyurular ile de desteklenmiştir.

https://spacechase.app/ bağlantısı aracılığıyla ulaşılabilen web sitesi, araştırma projesi hakkında bilgi vermek, geliştirilen yazılımları kullanmak/oynanmak üzere doğrudan ulaşımı sağlamak, bugüne dek yapılan atölyeleri ve bilimsel toplantı duyurularını arşivlemek, proje ekibine ait detaylara ve projeye ait iletişim bilgilerine yer vermek ve proje sürecinde ortaya çıkan yazılımlara ait sosyal medya hesaplarına erişimi sağlamak amaçları ile kurgulanmıştır.

Web sitesi, tasarım ve kodlama maliyetlerini azaltmak ve *hosting* hizmetlerini kolayca düzenleyebilmek adına web tasarım bilgisi gerektirmeden proje ekibi tarafından aynı anda

düzenlenebilme olanağı sağlayan *Wordpress* ara yüzü kullanılarak tasarlanmıştır. Web sitesi tasarımı aşamasında, Wordpress üzerinden "Premium" hizmet satın alınmış, bu hizmet alımı sonucunda site içeriğine uygun bir tema seçilmiş, site, içeriği kapsayacak sayfalara uygun düzen taslakları seçilip ihtiyaca uygun olacak şekilde düzenlenerek hazırlanmıştır. Hazırlanan sayfa taslaklarına içerikler yerleştirilmiş, sayfalara erişim sağlanabilmesi adına gezinme menüsü ve sosyal medya tuşları eklenmiştir. Wordpress arayüzünün kullanılmasının bir diğer avantajı da gezinme menüsü gibi özelliklerin programlama bilgisi gerektirmeden arayüz üzerinden kolayca üretilebilmesidir.

Web sitesinin güncel halinde:

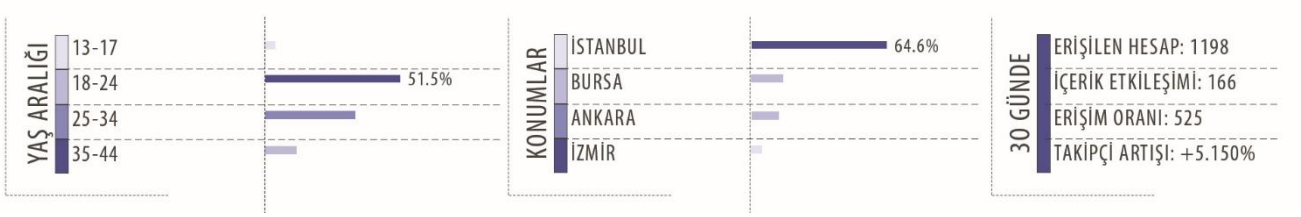
- Araştırmanın kapsam ve amacını anlatan, aynı zamanda anasayfa olan *"Home"*
- Yapılan atölye ve bilimsel toplantıların duyurularının arşivlendiği *"News"*
- Geliştirilen SpaceChase oyununun oynanabildiği *"Play"*
- Geliştirilen SpaceChase for Grasshopper yazılımının özelliklerini tanıtan, yazılımın kurulum ve kılavuz dosyalarını içeren *"SpaceChase for Grasshopper"*
- Proje ekibine ait bilgileri içeren ve ekibin bireysel iletişim bilgilerini sunan *"Team"*
- Proje ekibinin ortak e-posta adresine doğrudan mesaj atabilmeyi ve siteye abone olmayı sağlayan *"Contact"*

sayfaları yer almaktadır.

Oluşturulan web sitesinin yanı sıra proje kapsamında geliştirilen "SpaceChase" oyunu için Kasım-Aralık 2020 süresince, *700DCC Tasarım İletişim Danışmanlık* şirketi yardımıyla bilgi paylaşımı ve tanıtım çalışmaları yürütülmüştür. Amaç, pandemi sürecinde hızlı ve kolay anlaşılır şekilde proje hakkında bilgileri paylaşmak, etkinlik ve atölyeler için açık çağrıları yapmak ve proje ekibine ulaşılabilirliği sağlamaktır. Tanıtım ve iletişim çalışmaları kapsamında proje ve oyun hakkında tanıtım bülteni oluşturularak Aralık ve Ocak ayları içinde *Arkitera, Yapı Dergisi, Mimarizm, BT Güncel* gibi platformları da kapsayan çeşitli mimari, tasarım ve teknoloji içerikli basılı ve dijital medya yayınları ile paylaşılmıştır. Aynı zamanda SpaceChase oyununun tanıtımında hedef kitle olarak belirlenen mimarlık fakültesi öğrencilerine, akademisyenlere ve mimarlara yönelik olarak çeşitli sosyal medya platformları kullanılmıştır.

Seçilen kitleye en hızlı *instagram* platformu ile ulaşılmıştır. Instragram üzerinden ulaşılan hedef kitlenin %51 lik kısmını mimarlık fakültesinden, mimarlık, iç mimarlık ve şehir planlama bölümünden öğrenciler oluşturmaktadır (Şekil 59). *Linkedin* platformundan da mimarlık konusunda uzman profesyonellere ve akademisyenlere ulaşılmak üzere yararlanılmış, bu

platform verimli bir şekilde kullanılmıştır. Mevcut hedef kitle için *Facebook* platformunun çok tercih edilmediği gözlemlenmiştir.



Şekil 59. SpaceChase Game Instagram hesabı, Aralık ayı kullanıcı analizi.

Projeye süresince kullanılan ilgili sosyal medya hesaplarına aşağıda belirtilen linkler aracılığıyla ulaşılabilmektedir.

- Instagram: (spacechasegame) https://www.instagram.com/spacechasegame/?hl=tr
- (spacechaseforgrasshopper) https://www.instagram.com/spacechaseforgrasshopper/?hl=tr
- Linked In: (SpaceChase Game) https://www.linkedin.cn/in/spacechase-game-3904721bb/
- Facebook: (SpaceChase Game) https://www.facebook.com/SpaceChaseGame

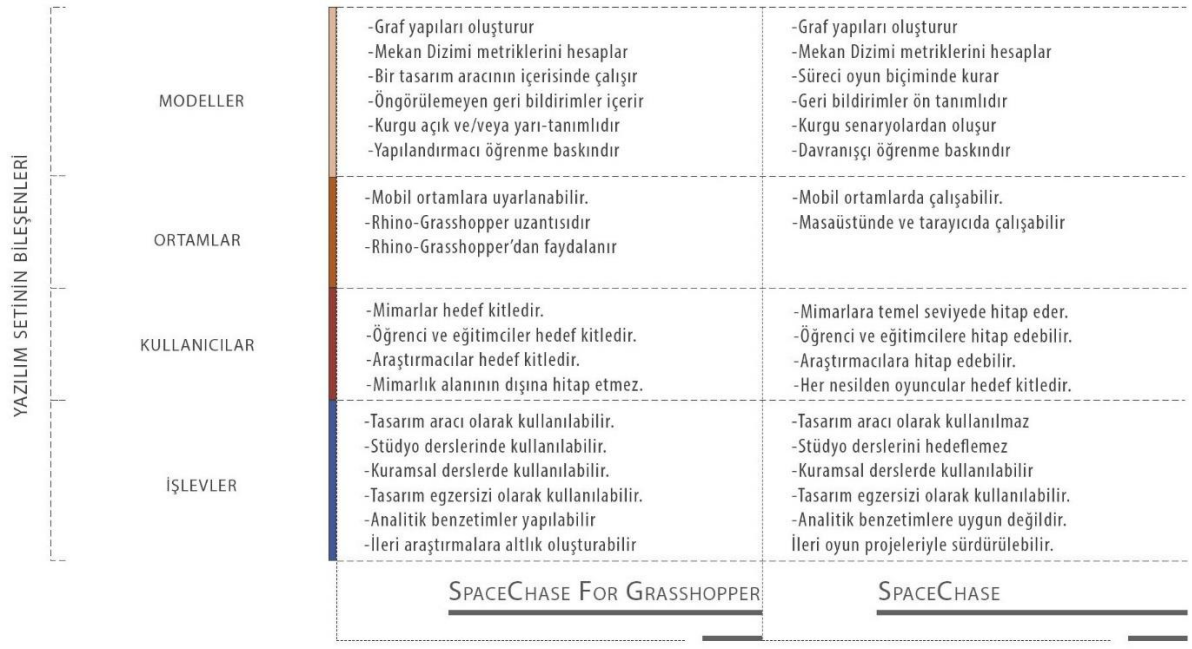
5. TARTIŞMA / SONUÇ

Bu bölüm öncelikle projenin çıktısı olan yazılımlara ilişkin karşılaştırmalı bir okumayı ele almakta, ardından gelecek çalışmalar için öneriler üzerine odaklanmaktadır.

5.1 Yazılımlara İlişkin Karşılaştırmalı Okumalar

Çalışma kapsamında Grasshopper eklentisi olarak çalışan oyunsal bir arayüzün (SpaceChase for Grasshopper) ve mimari anlamda oyunsal bir yazılım (SpaceChase)'ın eş zamanlı olarak birbirine paralel üretilmesi, iki yazılımın birbiri ile karşılaştırılmasını ve iki farklı bağlamın birbirinden öğrenmesini sağlamıştır. Grasshopper üzerinde çalışılması, bu yazılımın halihazırda mimarlık sektöründe ve eğitiminde yaygın olarak kullanılıyor olması, halihazırda benzer çalışmaların var olması, tasarımcının kendisinin kuracağı ilişkisel modelin değişkenlerini parametrize edebiliyor olması anlamında değerli bulunmuştur.

SpaceChase for Grasshopper (SCGH) ve SpaceChase (SC) detaylı bir şekilde karşılaştırıldığında şunlar söylenebilir: Her iki yazılım da genel olarak *mekansal ilişkileri modellemek ve analiz etmek, farklı dijital ortamlarda çalışmak, kullanıcı kitlesini çeşitlendirmek ve bütünleşik iş akışlarına katılmak* hedefleriyle üretilmiştir. Bu genel hedefler çerçevesinde yazılımların bileşenleri olarak; ***modeller, ortamlar, kullanıcılar, işlevler ve işleyiş mekanizmaları ve süreçler'***den söz edilebilir.



Şekil 60. Yazılım setlerinin bileşenleri.

5.1.1 Modeller

SCGH ve SC modelleri; "***analitik, dinamik ve etkileşimli"***dir. Aşağıda SCGH ve SC'in ürettiği modeller bu üç başlık altında incelenebilir.

Yazılımların ürettiği modellerin *analitik yapısı*, onların ağ temelli, hesaplamalı ve süreç odaklı olmalarını gerektirmiştir. Her iki yazılımda da ağ temelli bir altyapının bulunduğu gözlemlenmektedir. Her ikisinin de hesaplamalı doğaları gereği, Mekan Dizimi metrikleriyle çalışmaları mümkündür ve bu durum analitik modellerle çalışmalarını destekler ve güçlendirir. SCGH'ın modelleri, süreci bir tasarım aracının içerisinde gerçekleştirdiği için genel olarak tasarım süreci ile iç içe olma potansiyeli taşır. SC'in modelleri ise, süreci bir oyun biçiminde kurduğu için birer oyun sahnesinin parçasıdır.

Yazılımların ürettiği modellerin *dinamik yapısı*, onların hem hareketli hem de keşif odaklı olmalarını gerektirmiştir. Hem SCGH'ın, hem de SC'in hareket içeren modeller ürettiğini söylemek mümkündür. Konu sadece hareket bağlamında değil, hareketin doğurduğu keşif eylemi ile beraber düşünüldüğünde ise, SC'in keşfi tasarım faaliyeti ile bütünleştirmeye çalıştığı, SCGH'ın ise keşfi *oyunsallaşma* ile desteklediği görülmektedir. Bu ayrım SCGH ve SC arasındaki farkın netleşmeye başladığı noktadır.

Yazılımların ürettiği modellerin *etkileşimli yapısı*, onların geri bildirimli, kurgusal ve eğitsel olmalarını gerektirmiştir. SCGH'ın kullandığı arka plandaki optimizasyon algoritmaları nedeniyle etkileşim öngörülemeyen bir yapıdadır. SCGH ise, oyun sahneleri ile çalıştığı için öngörülemeyen bir geri bildirim içermez, etkileşimler ön tanımlıdır. Modellerin kurgusallığı, SCGH bağlamında açık veya yarı-tanımlı kurgulara olanak tanıyacak esnekliktedir. SCGH'da da tanımlı kurgular ve etkileşimler oluşturmak mümkün olmakla beraber bu durum genellikle kontrolden uzak ve hareket başladıktan sonra müdahale olanakları kısıtlıdır. Diğer yandan, SC, tümüyle kurgusallık üzerine kurulu olduğu için ön tanımlı sahneler yardımıyla etkileşim üzerinde tam bir kontrol sağlamak mümkündür. Her iki yazılımın sunduğu modellerin etkileşim özellikleri eğitsel bağlamda ele alındığında ise SCGH'ın yapılandırmacı bir kendi kendine öğrenme faaliyeti için daha uygun olduğu, SC'nin ise şartlayıcı bir davranışçı öğrenme ortamına daha uygun olduğu gözlemlenmektedir. SCGH'ın keşfe açık, yarı-tanımlı ve esnek yapısı onunla yapılabileceklerin tam olarak tanımlanmasının istenmediği eğitsel durumlar için idealdir. Diğer yandan, belirli sebep-sonuç ilişkilerinin vurgulanmak ve öğretilmek istendiği eğitsel durumlarda SC'nin sahne ve kurgu temelli yapısı daha uygundur.

5.1.2 Ortamlar ve Kullanıcılar

Her iki yazılımın çalıştığı ortamlar karşılaştırıldığında, SCGH'in mevcut durumda mobil ortamlarda çalışmaya uygun olmadığı görülmektedir. Arayüz özellikleri bakımından SC'nin ise mobil ortamlara daha uygun olduğu söylenebilir. Genel olarak, SCGH, belirli bir CAD yazılımı altında çalışmak üzere tasarlandığı için bu yazılımın olanaklarından faydalanmakta, bu yazılımın hedef kitlesinin etkin kullanımına açılmakta ve bu yazılımın ortamı ile sınırlı kalmaktadır. SC, masaüstünde ve tarayıcıda, başka bir yazılıma bağımlı olmadan çalışmaya uygundur. Bu ayrım, her iki yazılımın kullanıcı kitlesi arasındaki farkın netleştiği noktadır.

Oyunsal yazılımın mimarlar kadar mimarlık alanına ilgi duyan ancak mimar olmayan bir kullanıcı grubuna da hitap edebileceği düşünülmüştür; Grasshopper eklentisi ise doğrudan doğruya mimarlık öğrencileri, araştırmacılar, eğitmenler, pratik yapan mimarlara yöneliktir.



Şekil 61. Yazılım/ların hedefleri.

5.1.3 İşlevler

Proje kapsamında geliştirilen yazılımların ana işlevi, çeşitli ilişki matrisleri yardımıyla üretilen mekansal ağları ekran üzerinde ağ olarak görselleştirmek ve bu grafları Mekan Dizimi'nin kuram ve yöntemlerini kullanarak analiz etmektir. Bu işlev, doğrusal bir süreç içerisinde değil, grafların eş zamanlı olarak türetildiği ve analiz edildiği *dinamik* bir ortamda çalıştırılır.

Bu araçlar içinde SCGH, özellikle tasarım pratiğinde ve stüdyo derslerinde işlevselleşebilir kalitedir. SC ise mimarlık eğitimi içinde kuramsal tartışmalarda, küçük tasarım egzersizlerinde mekanın aklını, üretiliş ve deneyimleniş modellerini tartışmaya açmak için kullanılabilir.

5.1.4 İşleyiş Mekanizmaları ve Süreçler

Başlangıç düzeni ve veri girişi süreçleri incelendiğinde SCGH'da kullanıcının, ilk düğüm noktalarını ve bağlantılarını Rhinocheros ekranında yerleştirerek bir ağ diyagramı oluşturduğu ve bunları katman, isim ve öğe türü gibi özelliklerle etiketlediği söylenmelidir. Böylece eklentinin bunları kendi ekranında okuması mümkün kılınmaktadır. Bu eklenti, ikili bir excel dosyası üzerinden de çalışabilmektedir. SC'da ise sistem yöneticisi (admin), belirlenen sahneye göre ağ sınırlarını ayarlamak için her bir potansiyel düğüm noktasını ve bu noktalar aralarındaki tüm potansiyel bağlantıları bir excel dosyasına işler. Daha sonra sahnelere, bir dizi düğüm noktası ve bağ sınırlamasına ve bağlantı değişkenlerinin yinelenmesine olanak sağlayan benzer bir sayfadan geçirilir. Bu sahnelere veri girişi yapmayı sağlayan sayfalar, gerektiğinde ilk kurulumu geçersiz kılar. Açılış ekranı, oynatıcının sahneler arasında seçim yapmasını sağlar.

Arayüzde etkileşim bağlamında SCGH'da modelin yukarıda söz edildiği gibi, tasarım süreci boyunca hareket etmeye devam eden *dinamik* bir model olduğunun altı çizilebilir; mimar (kullanıcı) bağlar eklemeye ve silmeye devam edebilir, düğüm eklenmesi ve silinmesi durumunda ise dinamik ağ sıfırlanır ve yeniden oluşturulur. Dosya kaydedilebilir ve yeniden yüklenebilir bir formattadır. SC'da ise oturumlar (oyunun bölümleri) zaman açısından sınırlıdır (saniye veya dakika) ve bu sınırlama, sahnelere veri girişi yapmayı sağlayan excel sayfasında belirlenmektedir. Oyuncu, verilen düğüm noktaları takımından ekrana ekleme yapabilir, ekrandaki öğeleri birbiri ile bağlayabilir, bağlantısını kaldırabilir ve sürükleyebilir. Gelecekte, oyuncunun menüdeki düğüm noktaları takımında yer almayan düğümleri eklemesine de olanak sağlanması planlanmaktadır.

Çıktı ürün ve başarı bağlamında şunlar söylenebilir: SCGH'da kullanıcı, dosyayı kaydedebilmektedir ve .xls verilerini, aynı zamanda tekrarlı tasarım uygulamalarının başarısını/potansiyelini değerlendirebilen Mekan Dizimi analizlerini dışa aktarabilmektedir. Dosya kayıt edilebilmekte ve yeniden yüklenebilmektedir. Mekansal ağların kurulum sürecinde tek bir çözüm yolu bulunmamaktadır; tasarımcının ağ yapısını tekrarlı bir biçimde yapıp bozarak tasarım sürecini devam ettirebilmesine olanak sağlanmaktadır.

SC'in ise kullanıcının talep ederse kaydedebileceği ve paylaşabileceği bir formatta olması planlanmıştır. Mevcut durumda yazılım, kullanıcıya verilen süre içerisinde oyunun bitmesi yönünde kurgulanmıştır. Eğer düğüm noktası hiçbir bağlamsal düğüm noktasına bağlı değilse

oluşturulan mekansal konfigürasyonun bağlamla hiç ilişki kurmadığı kabul edilerek kullanıcının puanı düşürülmektedir. Ayrıca tasarlanan mekansal ağ kurgusu, sahneyi mümkün kılmak için metrik bir bakış açısından yetersiz kalırsa (örneğin, entegre olmayan sosyal mekansal öğe barındırmak gibi) veya programatik bir bakış açısından (konut sahnesinde kalmak) yetersiz kalırsa oyuncunun puanı düşer. Bu yazılımda elde edilen başarı, tek bir çözümden ziyade oluşturulan şemaların barındırdığı potansiyellere göre şekillenen bir durumdur.

Her iki yazılımın da düşündürücü ve eğlenceli olduğu düşünülmektedir. Her ikisi de önceden yüklenebilen örnek mekansal ağ kurgularından öğrenme sürecini beslemektedir. Öte yandan her iki yazılım da başlangıçta yorucu olarak nitelendirilecek bir girdi sağlanması sürecine ihtiyaç duymaktadır.

Düğümler arası ilişki dereceleri incelendiğinde SCGH'da düğüm noktası ile düğüm noktası veya bağlamsal düğüm noktası ve düğüm noktası arasındaki çekim gücünün mimar tarafından eklenti arayüzünde ayarlandığı ifade edilebilir. SC'da ise birimler arası çekim güçleri yönetici ana ekranında ayarlanabilmektedir. Gelecek çalışmalarda, oyuncunun bu güç ilişkilerini oyun sırasında değiştirebilmesi hedeflenmektedir. Oyun sırasında birimler arası ilişkiler, önceden belirlenmiş çekim kalitelerine göre oluşmaktadır. Gelecekte bu çekim kalitelerinin Mekan Dizimi metrikleri temel alınarak da belirlenmesi planlanmaktadır.

Düğümler arası çekim tipleri bağlamında şunlar söylenebilir: Her iki yazılımda da bağlam düğümler vardır, her iki arayüzde de birbirini çekme ve itme kaliteleri mümkündür. Oyunsal yazılımda birimler arası ilişkilerin dinamik yapısı kütle çekimini model alan bir yöntem ile hesaplanırken Grasshopper eklentisinde bu ilişkiler geometrik optimizasyon algoritmasına dayanan yay çekimleri (*spring*) ile modellenmektedir. Oyunsal yazılımda itme ve çekme etkileri yönetici ekranında tanımlanırken, Grasshopper eklentisinde itme ve çekme etkileri farklı sayısal girdilere izin vermektedir (Bkz. Bölüm 4.1). Her iki yazılımda hesaplama kolaylığı açısından düğüm noktası sayısı kısıtlı tutulmuştur; ancak her iki yazılımda da düğüm noktası sayısı arttırılarak daha karmaşık modellerin yapılabileceği düşünülmektedir.

Mekan Dizimi metrikleri / analizleri bağlamında iki yazılım karşılaştırıldığında her iki yazılımın da temel prensiplerinin ağ görselleştirilmesine ve tasarlanan bu ağ benzeri yapılara / mekanlara ilişkin sentaktik verilerin hesaplanmasına dayandığı vurgulanmalıdır. Mekanın bu ölçülen kaliteleri, Grasshopper eklentisinde daha okunabilir / görülebilir iken oyunsal yazılımda istendiğinde ulaşılabilen bir konumdadır. Her iki yazılımda da sadece program düğüm noktaları arasındaki *erişilebilirlik ilişkisi*, Mekan Dizimi hesaplamalarına dahil edilmekte, sentaktik hesap yapılmaktadır. Her iki yazılımda da bağlamsal düğüm noktaları mekansal özellik taşısa dahi bu hesaplara dahil edilmemiştir.

5.2 Öneri ve Değerlendirmeler

Mimari tasarım bilişsel bir aktivite olarak her zaman pratik ve teori üzerine kurulu bir mimarlık bilgisinden beslenmiş, bilimsel verilerin tasarım süreci içine nasıl taşınabileceği ise sorgulana gelmiştir. Bu araştırmanın ana ilgisi bilimsel veri ve süreçlerin tasarım süreci içine tasarımcının sezgilerini yönlendiren, düşüncelerini değerlendirmesine, onları yeniden strüktüre etmesine izin veren bir akılda nasıl dahil edilebileceği üzerindedir. Burada Simon (1969)'ın "doğal bilimler şeylerin nasıl olduğuyla, öte yandan tasarım nasıl olması gerektiği ile ilgilenir" şeklindeki keskin ayrımını bozmaya niyet edilerek bilimi, bilimsel verinin tasarımda "olmayanı" aramak için nasıl işlevselleştirebileceği araştırılmıştır. Bu sebeple Mekan Dizimi araştırmaları da bilimsel bir yaklaşım olarak böylesi bir sürece dahil olabilme olanakları ile değerlendirilmiştir. Ortaya koyduğu kuram ve araçlardan yola çıkılarak mimarlara tasarım sürecinde birlikte düşünebilecekleri yaratıcı, zihin açıcı, bir tasarım aracı geliştirilmeye çalışılmıştır. Süreç içinde kullanıcı profili geliştirilerek mimar olmayan kişilerin de deneyimleyebileceği oyunsal araçlar aranmıştır. Bu araç/lar Mekan Dizimi'nin mevcut bilgisayar temelli araçlarının olanak ve sınırlamaları üzerinden yapılan değerlendirmelerle geliştirilmiştir. Bu süreçte şu noktaların değerli olduğu düşünülmüştür.

- Mimari tasarım yapılarak öğrenilen bir *araştırma / keşif* sürecidir.

(olasılıkları arama / eskizler / denemeler / testler)

- Mimari tasarım tasarlanan ile tasarlayan arasında anda kurulan *dinamik bir etkileşim* süreci içinde gerçekleşir ve gelişir.

(tekrarlı denemeler / gerçek zamanlı geri bildirimler / veriler)

- Mimari tasarım belirli kurallar / sınırlar içinde ilerlemesi, tasarlayanın ana aktör olduğu bir süreçte denemelerle ilerlemesi, düşüncelerin anda temsil edilerek tartışmaya açılması pratikleri bağlamında *oyun benzeri* bir karakter göstermektedir.

(oyunsallık / kurallar)

Yapılan incelemeler mekansal kurguların nasıl kuruldukları ve nasıl deneyimlendiklerine ilişkin teorik bir kavrayış geliştirmeye niyet eden Mekan Dizimi araştırmalarının mimari tasarım sürecini bilimsel veri ile besleyen mimarlar için "onunla birlikte düşündüğümüz", yaratıcı, bilgi verici bir *tasarım aracı* olarak potansiyeller içerdiğini göstermiştir. Çalışmanın ana ilgisi bu doğrultuda mevcut araç setinin sadece analitik bir kavrayış geliştirmek yerine tasarımı an'da besleyen,

gerçek zamanlı bilgi üreten, dinamik ve etkileşimli bir araç setine nasıl evrilebileceği üzerine olmuştur.

Mekansal parçaları ve onlara eşlik eden bağlama ait verileri düğümler aracılığıyla temsil eden ve aralarındaki ilişkileri kurup bozarak olası bir araya gelmeleri, konfigürasyonları araştıran ve arka planda tüm olasılıkları ilgili Mekan Dizimi metrikleri ile değerlendirilebilir ve ölçülebilir hale getiren *SpaceChase* oyunu da bu doğrultuda geliştirilmiştir.

Araştırma sürecinde bu oyunsal yazılıma bir ön hazırlık yapılması gerekli görülmüş, Mekan Dizimi analizleri için ilgili algoritmaların geliştirildiği, dinamik ve etkileşimli ağ yapılarının kurulmasına ve analizine olanak veren bir yazılım eklentisi, SpaceChase for Grasshopper, geliştirilmiştir. Süreç içinde verileriyle oyunsal yazılımı besleyeceği düşünülen bu çalışma değerli bulgularıyla planlananın ötesinde derinleştirilmiş, gelecek araştırmalara da konu edilebilecek bir olgunluğa ulaştırılabilmiştir.

Özetle bu araçlar yardımıyla Mekan Dizimi araştırmalarının tasarım süreci içinde kullanılabilirliği;

- tanımlayıcı (descriptive) olandan, yaratıcı (generative) olana
- statik olandan, dinamik ve etkileşimli olana
- son ürünü anlatandan, tasarım süreci içinde olasılıkları arayana
- bitmiş veri ve haritalamadan, verili data ile tasarımcı tarafından manipüle edilebilen kişisel anlatılara
- sadece mekansal olandan aynı zamanda bağlamsal da olana

dönüştürülmeye çalışılmıştır.

Sözü edilen araçlar kuramdan pratiğe çok disiplinli bir çalışma ekibinin ürünüdür. Bu tür bir ekip çalışması araçların kalitelerini çok farklı açılardan araştırma ve sorgulama olanağı doğurmuştur. Benzer şekilde araştırmanın ana argümanı ve geliştirdiği yazım/lar farklı mimarlık okullarından, farklı disiplinlerden öğrenciler, araştırmacılar, eğitimciler ile ulusal ve uluslararası platformlarda (atölyeler, bilimsel toplantılar) tartışılabilmiş, tüm veriler araştırmanın derinleştirilmesine, yazılım/ların güncellenmesine olanak vermiştir. Elbette yaşanan gelişmeler araştırmanın ilerleyişinde / yürütülmesinde zorluklar doğurmuş, etkinliklerinin (araştırma toplantıları, yazılım geliştirme çalışmaları, yazılım testleri, atölyeler, bilimsel toplantılar, vb.) dijital ortama taşınmasını gerekli kılmıştır. Ortaya çıkan koşullar üretimlerin test edilmesi, ulusal ve uluslararası platformlarda paylaşılması ve tartışılması anlamında da yeni potansiyeller yaratmıştır. Bu öngörülemeyen durum kapsamında araştırmanın belirlenen hedefler içinde tamamlandığı söylenmelidir.

Araştırmanın özetle şu alanlarda değer ürettiği düşünülmektedir:

- Mekan Dizimi araştırmalarının / araçlarının zenginleştirilmesi

- Var olan araç setinin sadece analitik olmak ötesinde tasarımcıyla karşılıklı diyalog/etkileşim kuran ve kararlarının mekansal kalitelerini gerçek zamanlı ölçebilen *dinamik, etkileşimli* araç setine evrilme potansiyelinin araştırılması
- Mekansal ağların ürettiği kaliteleri kavrama / çözümleme sürecinde mekanın içinde bulunduğu bağlama ait verilerin değerlendirilme olasılıklarının araştırılması
- İtme ve çekme kaliteleri ile mekanlar arası gruplaşma / kotlara ayrılma potansiyellerinin araştırılması
- Kullanıcı deneyimi üzerinden mekana ilişkin işlenebilecek kapsamlı bir veri tabanının elde edilebilmesi / bir katalog oluşturulabilmesi
- Özellikle oyun bağlamında basit bir dil sunulabilmesi, aracın erişilebilir, kullanılabilir olma kalitesinin çoğaltılması

- Mimari tasarım sürecinin zenginleştirilmesi

- Yaparak öğrenme pratiği içinde eskiz benzeri bir kalite ile bilimsel verinin sürece dahil edilmesi
- Oyunlaşma (yapma / yeniden yapma) hali üzerinden tasarım kararlarının / olasılıklarının araştırılması
- Dijital arayüzlerin / araçların sonuç ürünü temsil etmenin ötesinde bu ürünü aramada yaratıcı bir şekilde sürece dahil edilmesi

- Mimari eğitimde öğrenme pratiklerinin zenginleştirilmesi

- Mekana ilişkin farkındalığın (mimari program / yaşam senaryosu / beden/lerin hareketi / rotaları / etkinlikleri / mekanın hikayesi, tasarlanma ve çalışma prensipleri / tasarlanmış mekanın potansiyelleri) arttırılması
- Bilimsel verinin tasarım sürecindeki değerinin vurgulanması

- Mimar olmayan kullanıcı grubu için bir öğrenme ve farkındalık ortamı kurulması

- Mekanın hikayesi ile birlikte nasıl kurgulanıp çalıştığı ile ilgili bir kavrayışın oyunsal bir ortamda izlenmesi / düşünülmesi
- Farklı kullanıcı gruplarının hikayelerinin ve mekansal dizilim tercihlerinin oyun kayıtları üzerinden izlenebilir olması ile stüdyo benzeri “birbirinden öğrenme ortamları”nın yaratılması

Araştırmanın gelecekte şu yönlerde geliştirilebileceği düşünülmektedir:

Yazılımlara ilişkin olarak;

- Mekan Dizimi ölçümleri zenginleştirilerek, ölçüm metrikleri çoğaltılabilir. İtme ve çekme kalitelerini ve de senaryo hedeflerini belirleyecek şekilde yazılımlar geliştirilebilir. Dinamik ağ yapısı, Mekan Dizimi metriklerinin hareket esnasında okunabileceği bir özellikle ele alınabilir.

- Mekan Dizimi analizlerinde mekanlar arası girilebilirlik ilişkileri yanında görülebilirlik kaliteleri de araştırma konusu yapılabilir.
- Üçüncü boyut ve geometrik kaliteler sürece dahil edilebilir.
- Tasarlanan mekanın ölçeği (düğüm sayıları) ve bağlamı (kent) büyüyebilir.
- Çok oyunculu bir oyun mekaniği geliştirilebilir.
- Açık uçlu “açık dünya” oyun mekanikleri ile tasarım sürecine yakınlaşan bir oyun mekaniği geliştirilmesi için araştırma yapılabilir.
- Oyun seanslarının birikimi ile mekansal ilişkilenme ile sosyal fiziksel ilişkilenme üzerine teorik araştırma yapılabilir.

Sürece ilişkin olarak;

- Kurumsal yapıda, üniversiteler bünyesinde ulusal ve uluslararası düzeyde çeşitli araştırma laboratuvarları kurularak çok disiplinli olarak süreç işletilebilir.
- Kuramsal alt yapının ve üretilen yazılımların mimari tasarım eğitiminde daha kapsamlı ve sistematik bir şekilde ele alınması (örneğin bir ders / sempozyum), deneylerin çoğaltılması, geri bildirimlerle yazılımlara ilişkin güncellemelerin yapılması, yeni açılımlar yapılması konularına odaklanılabilir.

KAYNAKLAR

Benedikt, M. L. 1979. “To take hold of space: isovists and isovist fields”, Environment and Planning B: Planning and design, 6(1), 47-65.

Bielik, M., Koenig, R., Schneider, S. 2012. “Parametric Urban Patterns: Exploring and integrating graph-based spatial properties in parametric urban modelling”, Proceedings of eCAADe 2012.

Cooper, C., Chiaradia, A. J. 2015. “sDNA: how and why we reinvented Spatial Network Analysis for health, economics and active modes of transport”, Proceedings of GISRUK 2015. School of Geography, University of Leeds.

Corrodi M. 2006. “On the Kitchen and Vulgar Odors”. In: Spechtenhauser K. (eds) The Kitchen. Living Concepts, vol 1. Birkhäuser Basel, 21-43

Cross, N. 2001. “Designerly Ways of Knowing: Design Discipline Versus Design Science”, Design Issues, 17(3), 49-55

Cross, N. 2007. Designerly Ways of Knowing Board of International Research in Design, Birkhäuser Architecture, Basel: Boston, Berlin.

Çatak, G. 2011. “Oynarken tasarlamak: Dijital tasarım oyunları”, Sigma, 3, 385-391.

Derix, C. 2014. “The Space of People in Computation”, in Derix, C and Izaki, A (eds) 2014, Empathic Space: The computation of human-centric architecture, Architectural Design, John Wiley & Sons.

Derix, C., Izaki, Å. 2013. “Spatial Computing for the New Organic”, Architectural Design, 83, 42-47.

Dijkstra, E. W. 1959. “A Note on Two Problems in Connexion with Graphs”, Numerische Mathematik, 1(1)1, 269–271. doi:10.1007/BF01386390.

Dursun, P. 2007. “Space Syntax in Architectural Design”, Proceedings of 6th International Space Syntax Symposium, ITU Faculty of Architecture, 12-15 Haziran 2007, Istanbul.

Dursun Çebi, P., Kozikoğlu, N. 2017. “Network Thinking Architectural Design: Experimenting and Working with Relational Models”, Proceedings of 11th International Space Syntax Symposium, Lisbon, 1- 10.

Dorts, K. 2010. “The Nature of Design Thinking”, Design Thinking Research Symposium, Faculty of Design, Architecture and Building, University of Technology, Sydney, October 2010 in DTRS8 Interpreting Design Thinking: Design Thinking Research Symposium Proceedings, ed. Dorst, K., Stewart, S., Staudinger, I., Paton, B., Dong, A., DAB Documents, Sydney, Australia, 131-139.

Feng, C., Zhang, W. 2017. “Grasshopper Reach Analysis Toolkit: Interactive parametric syntactic analysis”, Proceedings of the 11th Space Syntax Symposium.

Gil, J., Varoudis, T., Karimi, K., Penn, A. 2015. “The Space Syntax Toolkit: integrating depthmapX and exploratory spatial analysis workflows in QGIS”, Proceedings of the 10th International Space Syntax Symposium.

Hanson, J. 2003. “Decoding Homes and Houses”, Cambridge University Press.

Hillier, B. 1996. “Space is the Machine”, Cambridge: Cambridge University Press.

Hillier, B., Hanson, J. 1984. “The Social Logic of Space” Cambridge University Press.

Hillier, B., Burdett, R., Peponis, J., Penn, A. 1987. “Creating Life: Or, Does Architecture Determine Anything?”, Architecture & Comportement/Architecture and Behaviour, 3 (3), 233 - 250

Hillier, B., Hanson, J. 1997. “The Reasoning Art: or, The Need for an Analytical Theory of Architecture”, Proceedings of 1st International Space Syntax Symposium, UCL, London.

Huizinga, J. 2020. Homo ludens. Editora Perspectiva SA.

Iida, S. 2001. “Segmen”, UCL, London.

Klarqvist, B. 1993. “A Space Syntax Glossary”, Nordisk Arkitekturforskning. 1993:2, 11-12.

Koch, D. 2004. “Spatial Systems as Producers of Meaning: The idea of knowledge in three public libraries”, Licentiate thesis. Stockholm: KTH School of Architecture/Spatial Analysis.

Kozikoğlu, N., Dursun Çebi, P. 2015. “Thinking and Designing with the Idea of Network in Architecture”, AZ, 2(3), 71-87.

Lawson, B. 2005. How Designers Think (4. Basım). Burlington: Architectural Press, Elsevier.

Mcgaughey, S. 2017. “Kitchen Stories: Literary and Architectural Reflections on Modern Kitchens in Central Europe”. Crossing Central Europe Continuities and Transformations, 1900 and 2000, University of Toronto Press, 100-118

Markhede, H., Miranda Carranza, P. 2007. “Spatial Positioning Tool: a prototype software and some background correlation data”. Proceedings of 6th International Space Syntax Symposium, Istanbul (Vol. 2, 102-1).

Nourian, P., Rezvani, S., Sariyildiz, S. 2013a. “A Syntactic Architectural Design Methodology”, Proceedings of 9th International Space Syntax Symposium, Seoul, Korea.

Nourian, P., Rezvani, S., Sariyildiz, S. 2013b. “Designing with Space Syntax, A configurative approach to architectural layout, proposing a computational methodology”, Computation and Performance, eCAADe 31, Spatial Performance and Space Syntax – Vol. 1, 357-365.

Nourian, P. 2016. Configraphics: Graph Theoretical Methods for Design and Analysis of Spatial Configurations. Dissertation for obtaining the degree of doctor at Delft University of Technology.

Rodriguez, H. 2006. “The Playful and the Serious: An approximation to Huizinga's Homo Ludens”, The International Journal of Computer Game Research, 6(1), Aralık 2006.

Schaffranek, R., Vasku, M. 2013. “Space Syntax for Generative Design: on the application of a new tool”, Proceedings of the 9th International Space Syntax Symposium.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., ... & Ideker, T. (2003). “Cytoscape: a software environment for integrated models of biomolecular interaction networks”. Genome Research, 13(11), 2498-2504.

Schön, A. D. 1987. Educating the Reflective Practitioner. Jossey-Bass, A Wiley Imprint: San Francisco.

Sevtsuk, A., Mekonnen, M. 2012. “Urban network analysis: a new toolbox for measuring city form in ArcGIS”. Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design,1-10.

Simon, H. A. 1996. The Sciences of the Artificial. MIT Press, Cambridge, Massachusetts, London, England.

Turner, A. (eds) 2007. New developments in Space Syntax Software, 6th International Space Syntax Symposium, Istanbul Technical University: Istanbul, Turkey.

Turner, A., Penn, A., Hillier, B. 2005. “An algorithmic definition of the axial map”. Environment and Planning B: Planning and Design, 32(3), 425-444.

Yılmaz, E. A. 2016. Oyunlaştırma, Abaküs, İstanbul

URL-1: https://www.spacesyntax.net/software/

URL-2: https://unity.com/

EKLER

EK 1. SpaceChase for Grasshopper WIP Prototip #2 Veri Akış Diyagramı

EK 2. SpaceChase for Grasshopper WIP Prototip #3 Veri Akış Diyagramı

EK 3. SpaceChase for Grasshopper WIP Prototip #4 Veri Akış Diyagramı

EK 4. SpaceChase for Grasshopper WIP Prototip #4 Ekranları

EK 5. SpaceChase for Grasshopper Beta (0.1.0) Geliştirilen Bileşenler

EK 6. SpaceChase for Grasshopper Beta (0.1.0) Örnek Veri Akış Diyagramı

EK 7. SpaceChase for Grasshopper Beta (0.2.5) Geliştirilen Bileşenler

EK 8. SpaceChase for Grasshopper Beta (0.2.5) Örnek Veri Akış Diyagramları

EK 9. SpaceChase for Grasshopper Beta (0.3.3) Geliştirilen Bileşenler

EK 10. SpaceChase for Grasshopper Beta (0.3.3) Örnek Veri Akış Diyagramları

EK 11. SpaceChase for Grasshopper Beta (0.3.3) Kullanım Kılavuzu

EK 12. SpaceChase for Grasshopper Değişim Günlüğü (*Chagelog*)

EK 13. SpaceChase Oyunu Arayüz Etkileşimini Gösteren Örnek Ekran Görüntüleri

EK 14. Oyun Elemanları İlişkilerinin Anlatım Diyagramı

EK 15. SpaceChase Oyunu Kullanım Kılavuzu

EK 16. Atölye I Programı

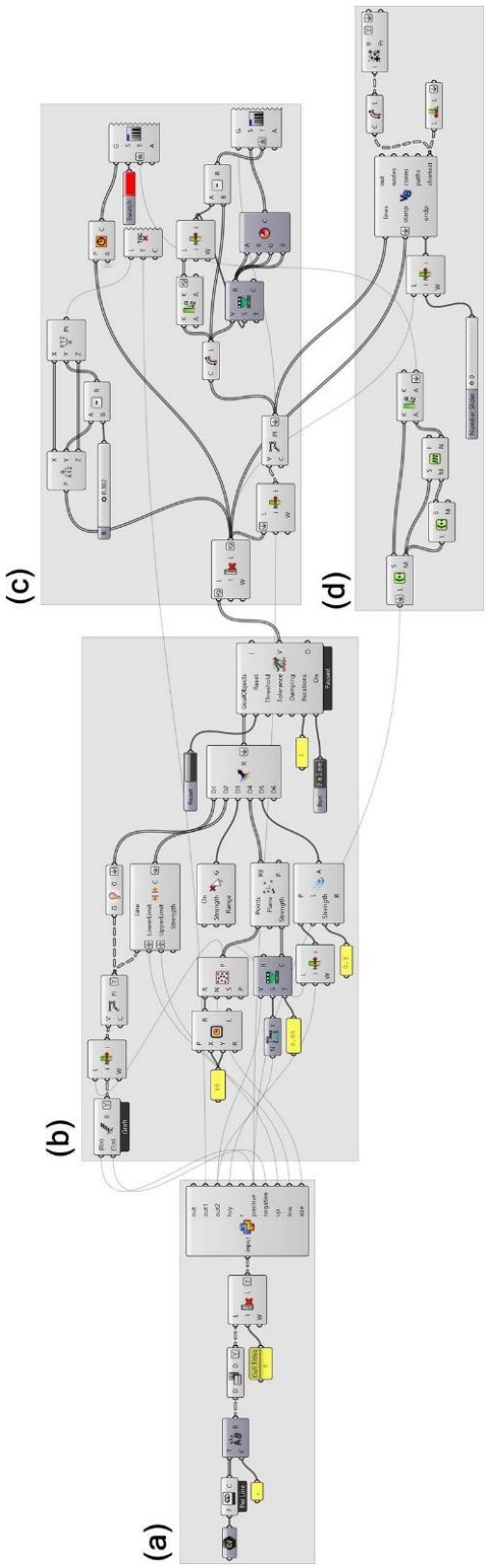
EK 17. Atölye II Programı

EK 18. Atölye III Programı

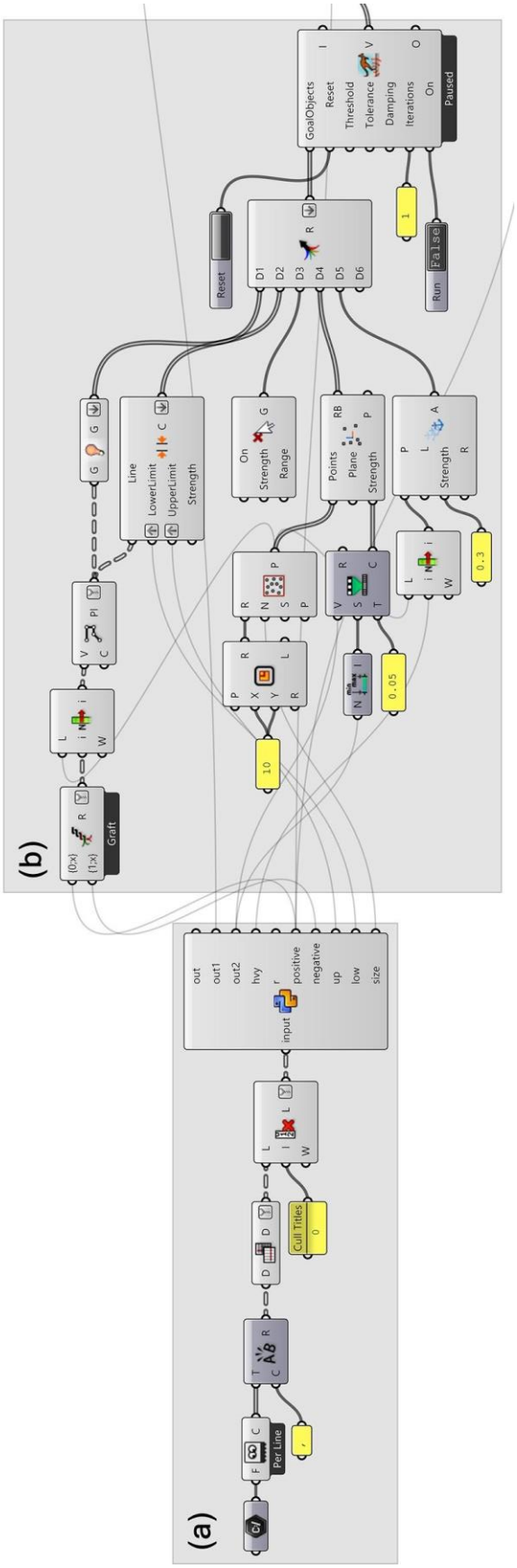
EK 19. Atölye IV-V Programı

EK 20. Terimler Dizini

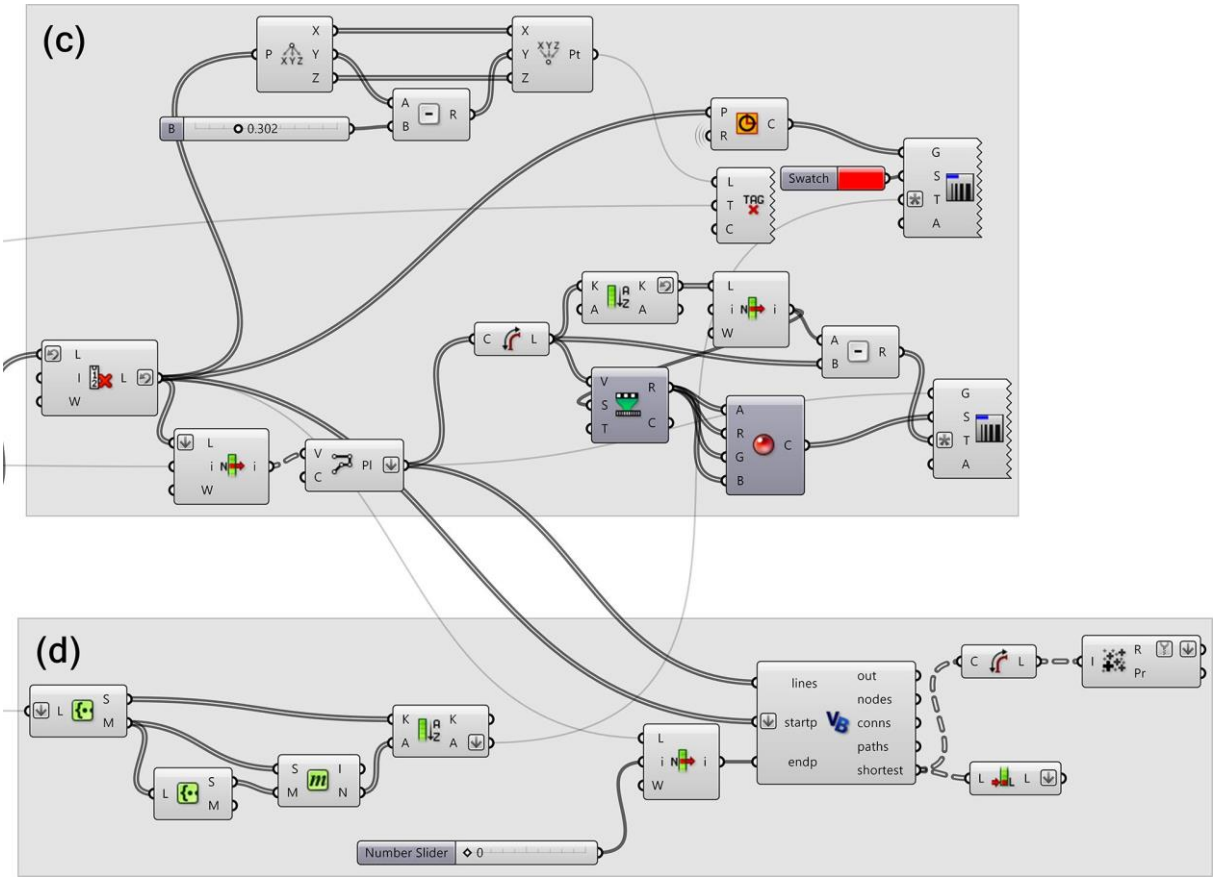
EK 1. SpaceChase for Grasshopper WIP Prototip #2 Veri Akış Diyagramı



Şekil Ek 1-1. Prototip #2 veri akış diyagramı: (a), (b), (c) ve (d) gruplarının genel görünümü.

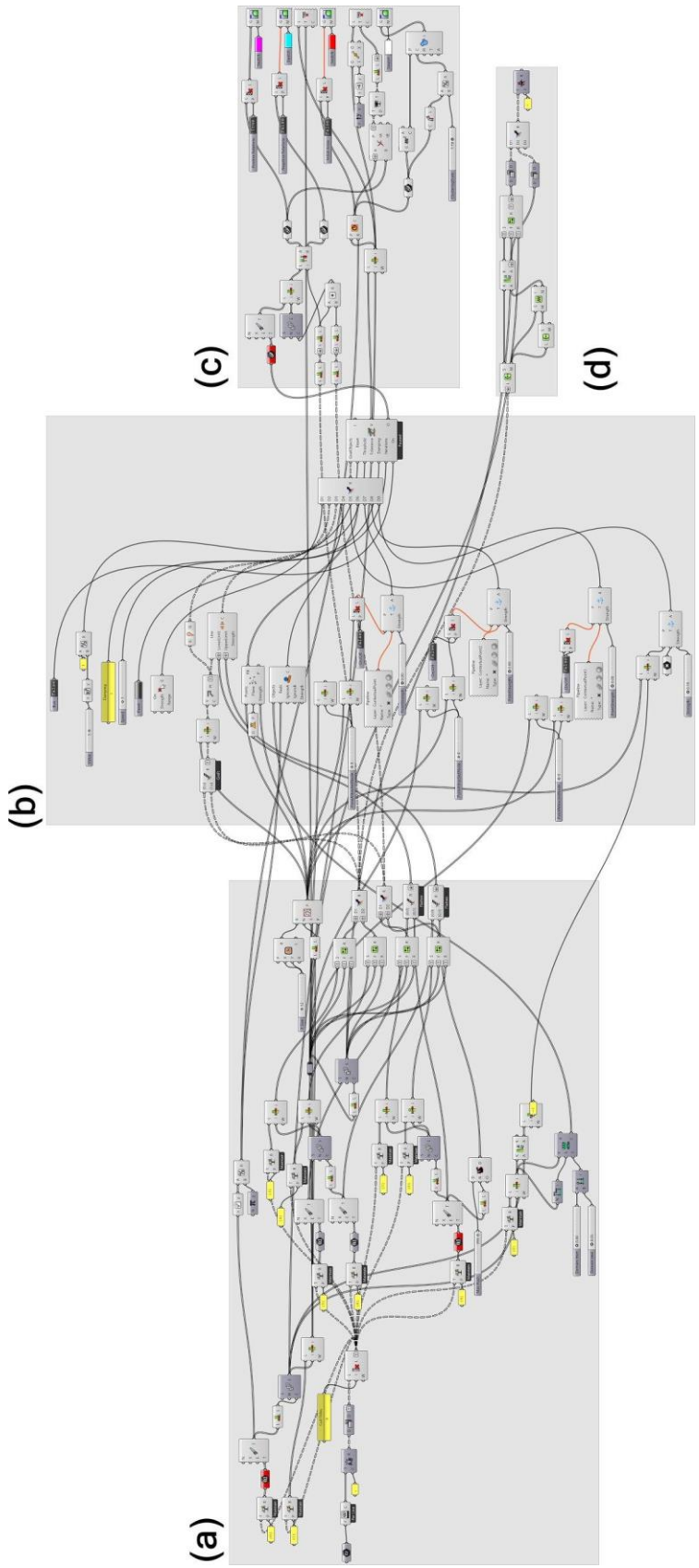


Şekil Ek 1-2. Prototip #2 veri akış diyagramı: (a) ve (b) gruplarının detayı.

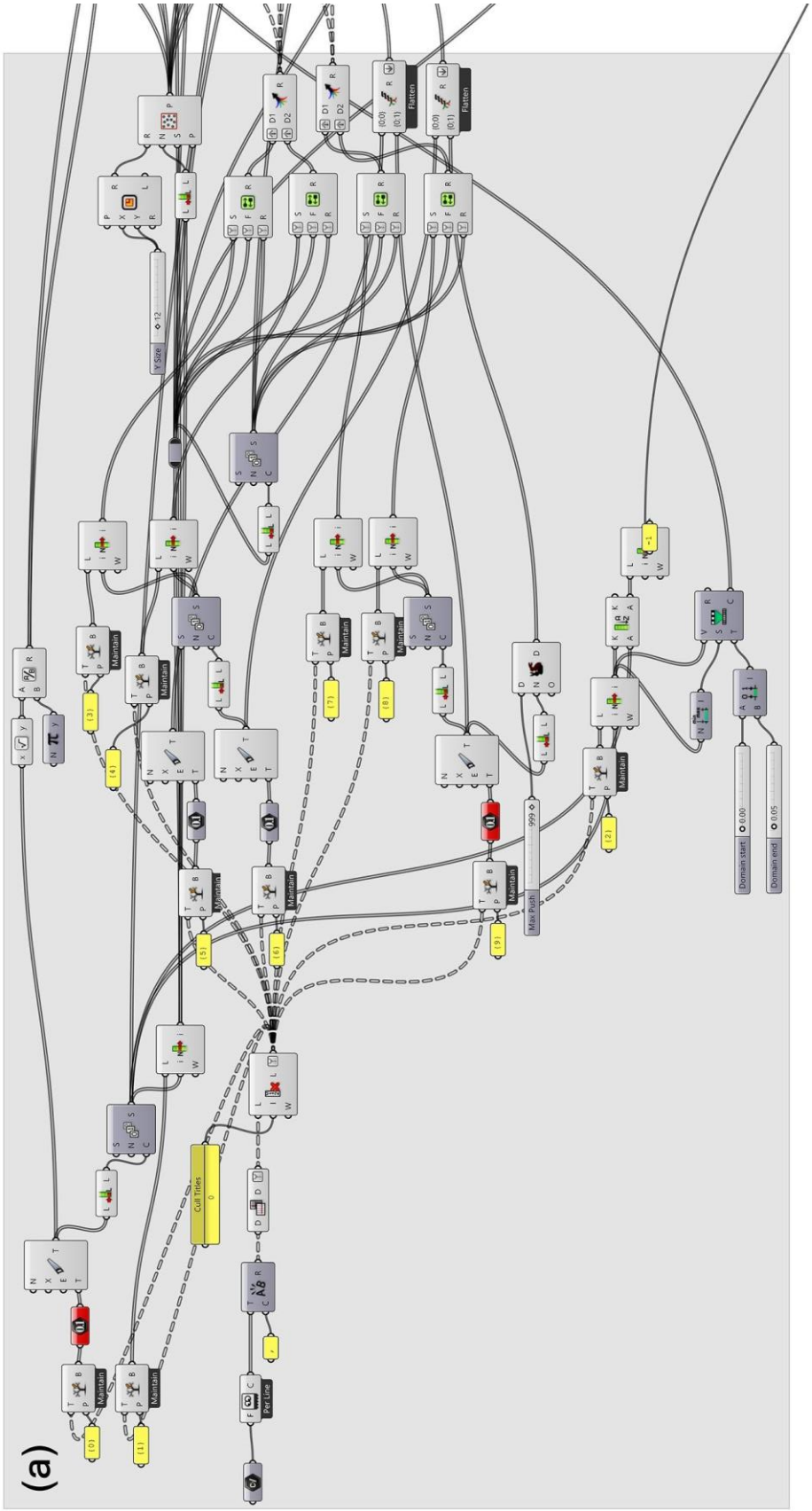


Şekil Ek 1-3. Prototip #2 veri akış diyagramı: (c) ve (d) gruplarının detayı.

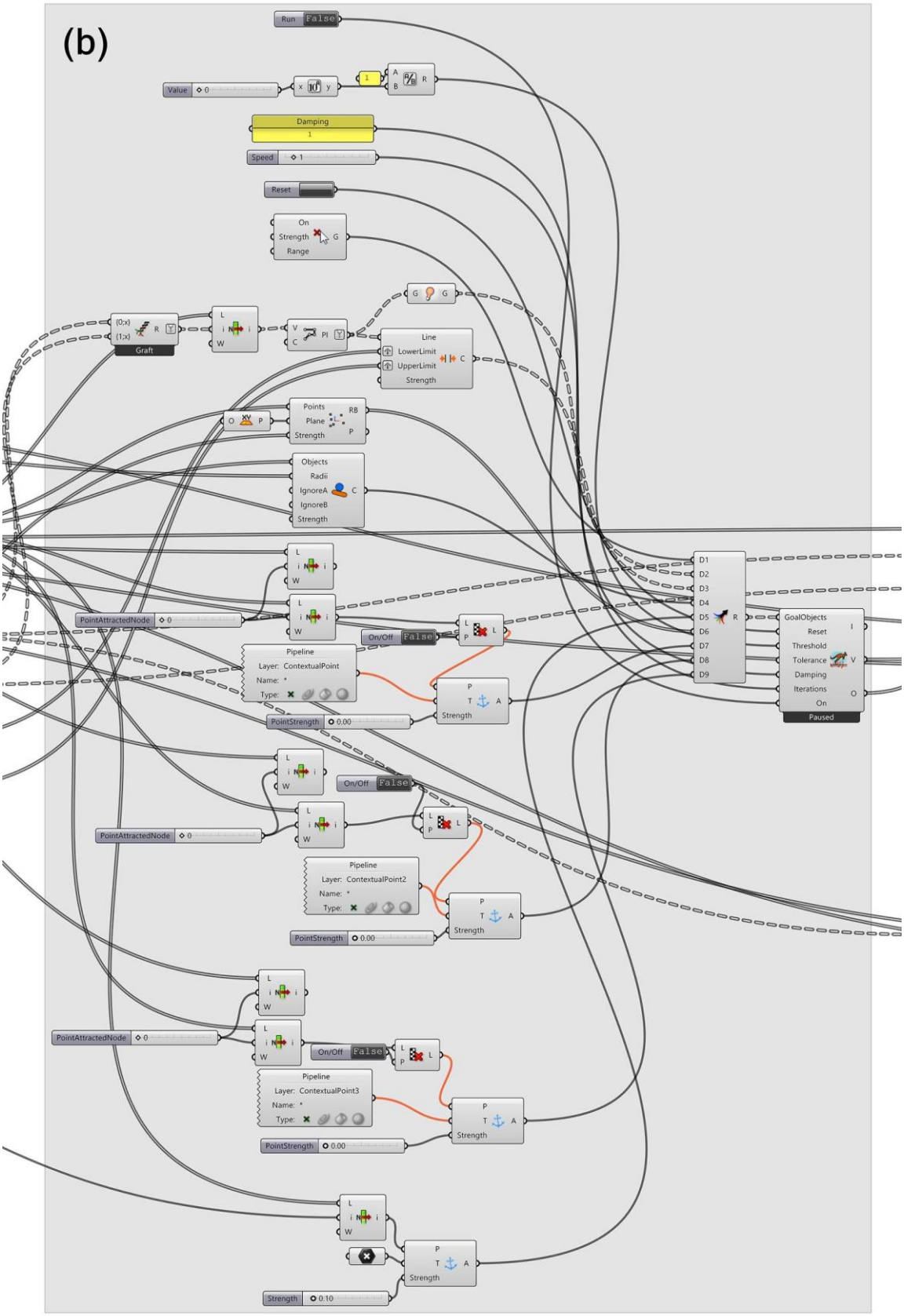
EK 2. SpaceChase for Grasshopper WIP Prototip #3 Veri Akış Diyagramı



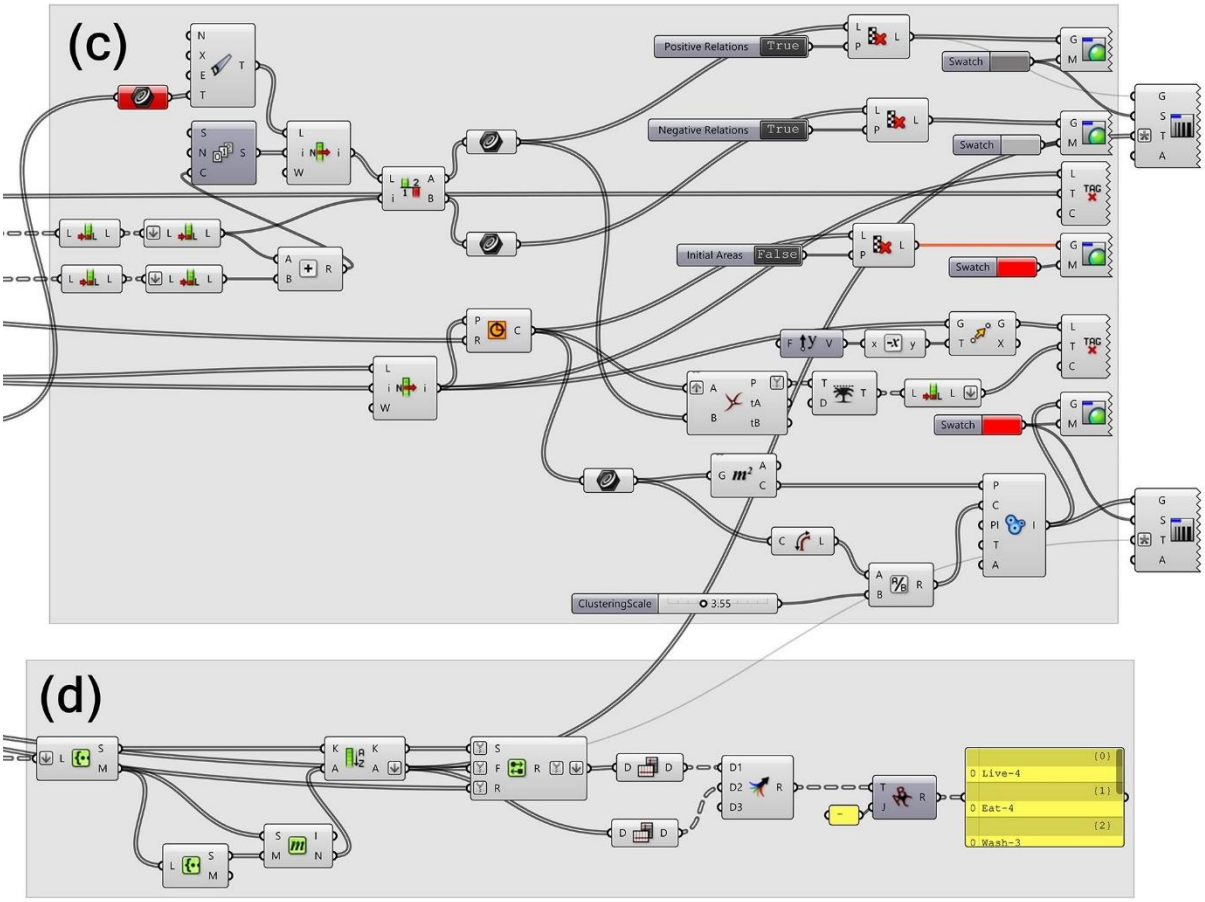
Şekil Ek 2-1. Prototip #3 veri akış diyagramı: (a), (b), (c), ve (d) gruplarının genel görünümü.



Şekil Ek 2-2. Prototip #3 veri akış diyagramı: (a) grubunun detayı.

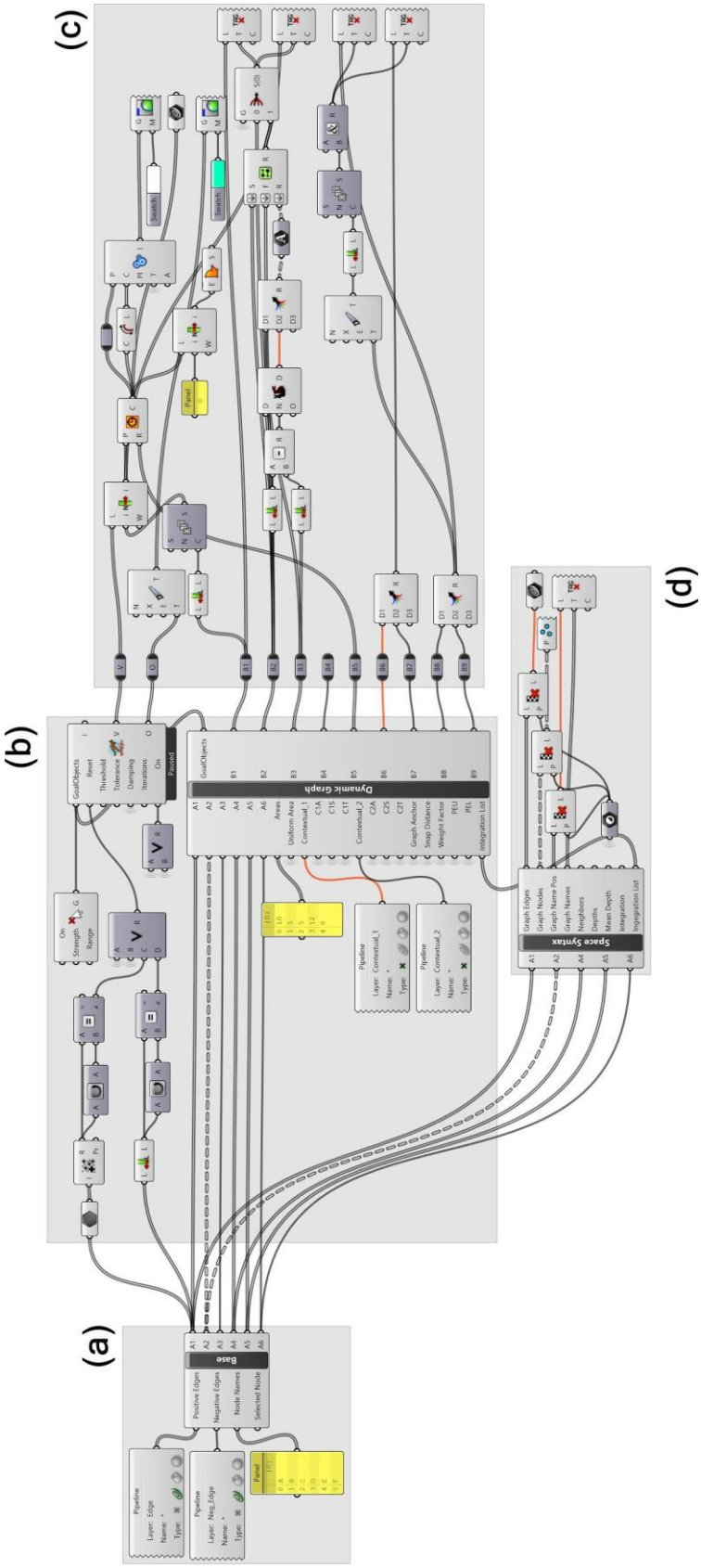


Şekil Ek 2-3. Prototip #3 veri akış diyagramı: (b) grubunun detayı.

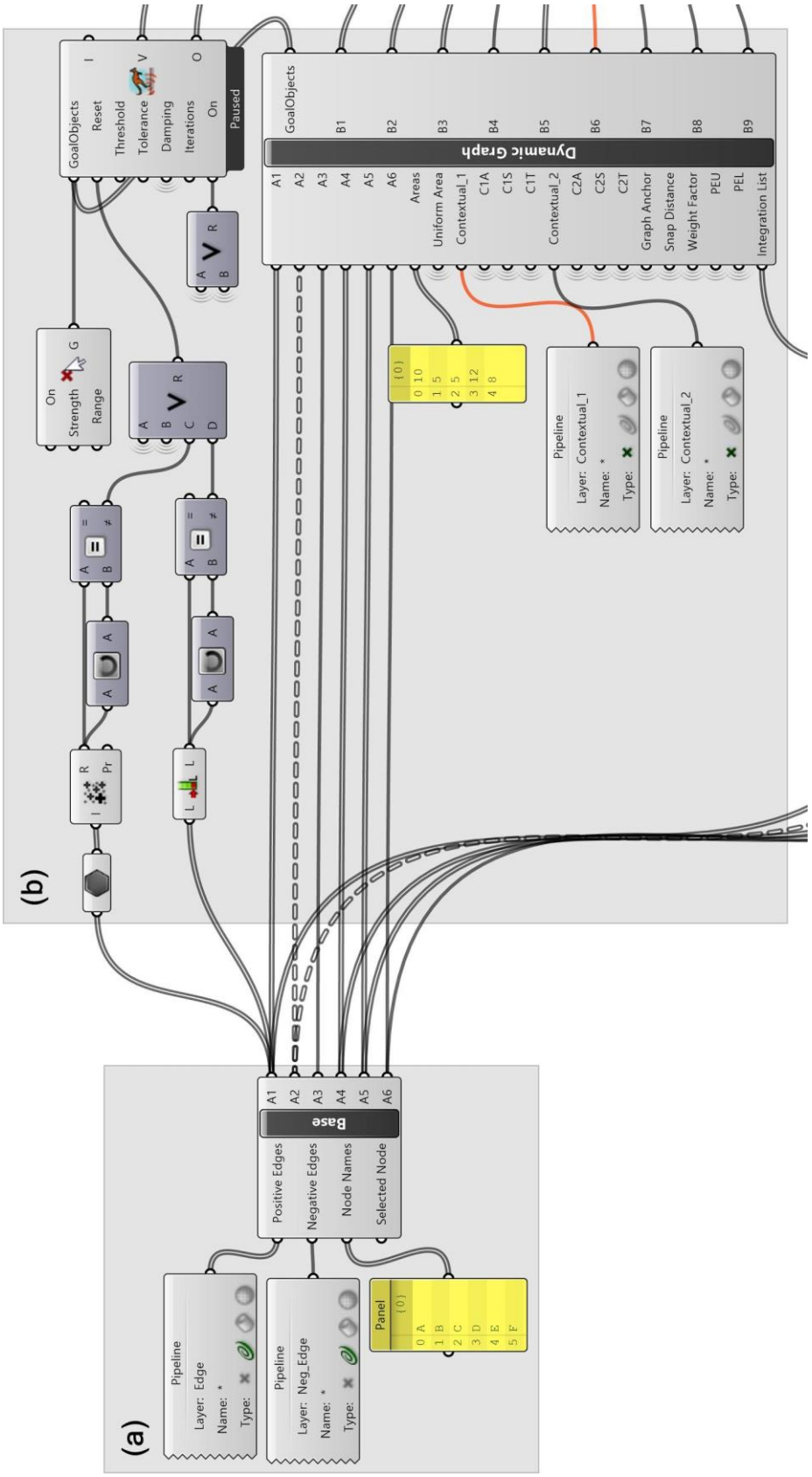


Şekil Ek 2-4. Prototip #3 veri akış diyagramı: (c) ve (d) gruplarının detayı.

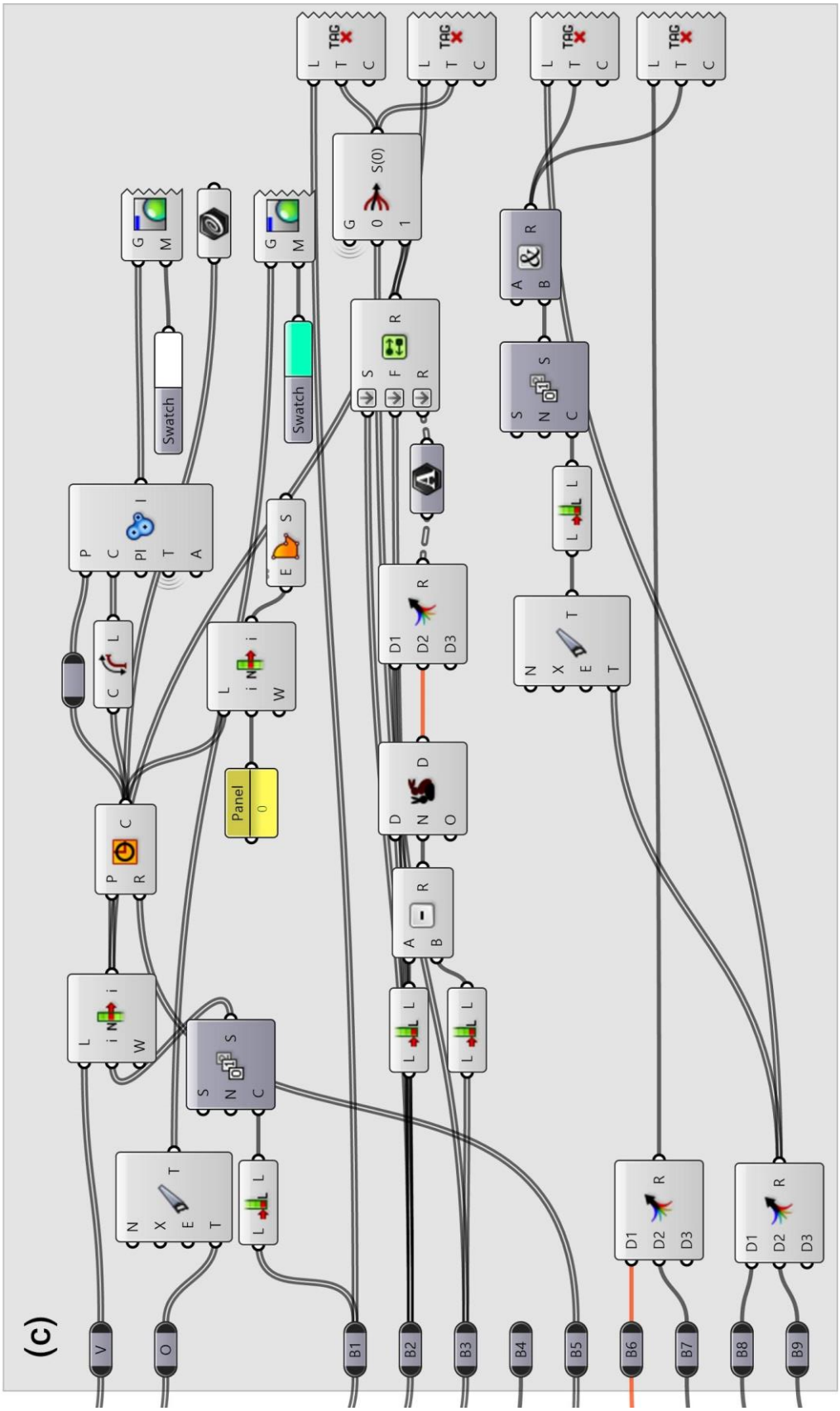
EK 3. SpaceChase for Grasshopper WIP Prototip #4 Veri Akış Diyagramı



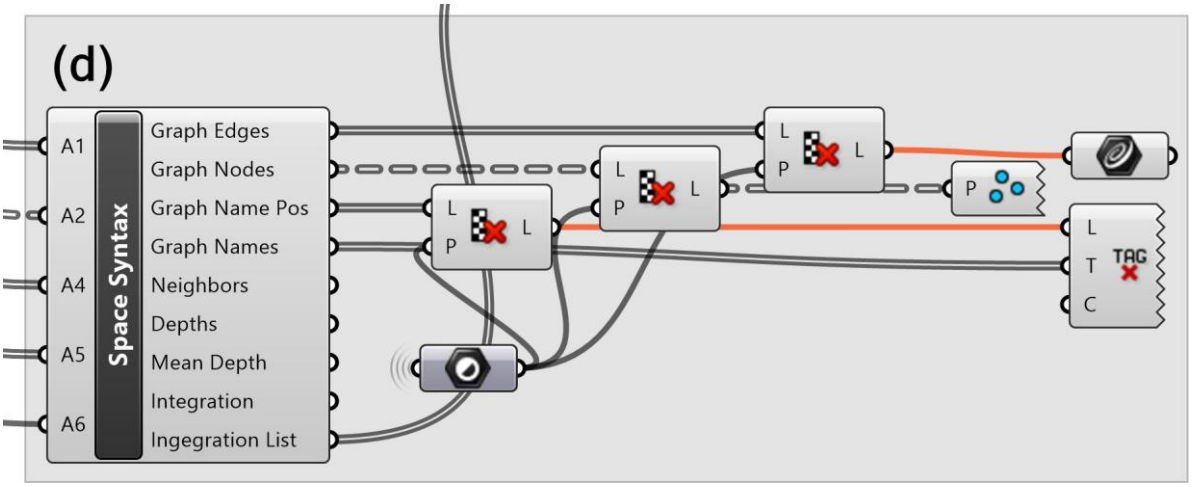
Şekil Ek 3-1. Prototip #4 Veri akış diyagramı: (a), (b), (c), ve (d) gruplarının genel görünümü.



Şekil Ek 3-2. Prototip #4 veri akış diyagramı: (a) ve (b) gruplarının detayı.

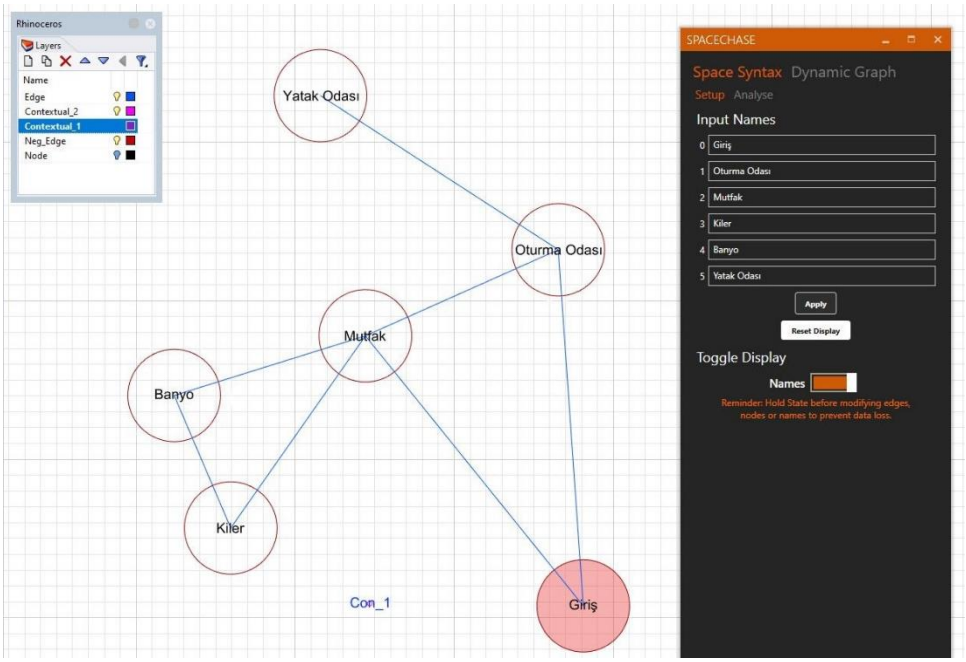


Şekil Ek 3-3. Prototip #4 veri akış diyagramı: (c) grubunun detayı.

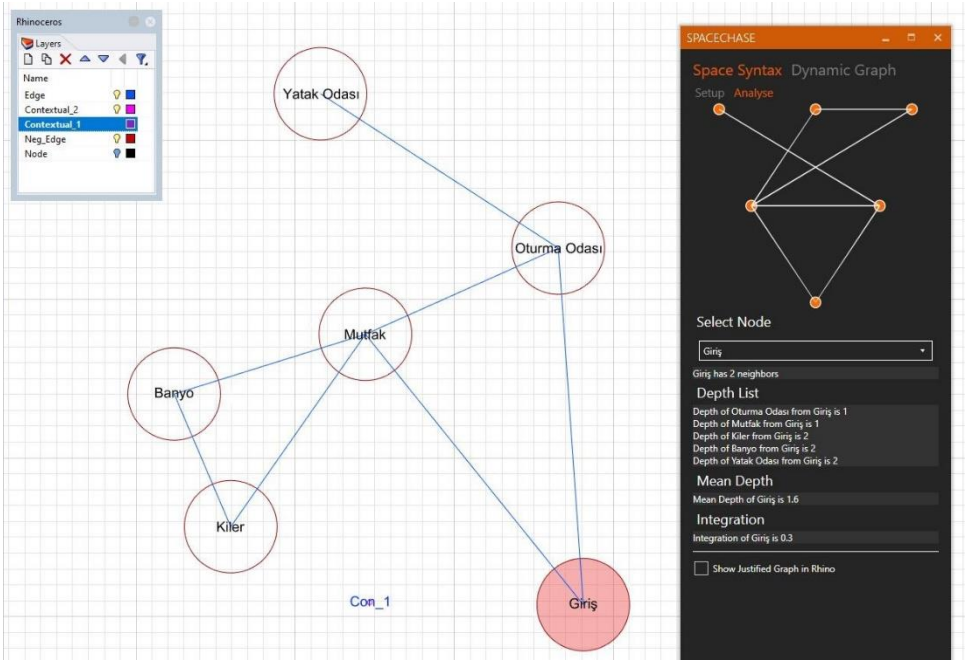


Şekil Ek 3-4. Prototip #4 veri akış diyagramı: (d) grubunun detayı.

EK 4. SpaceChase for Grasshopper WIP Prototip #4 Ekranları

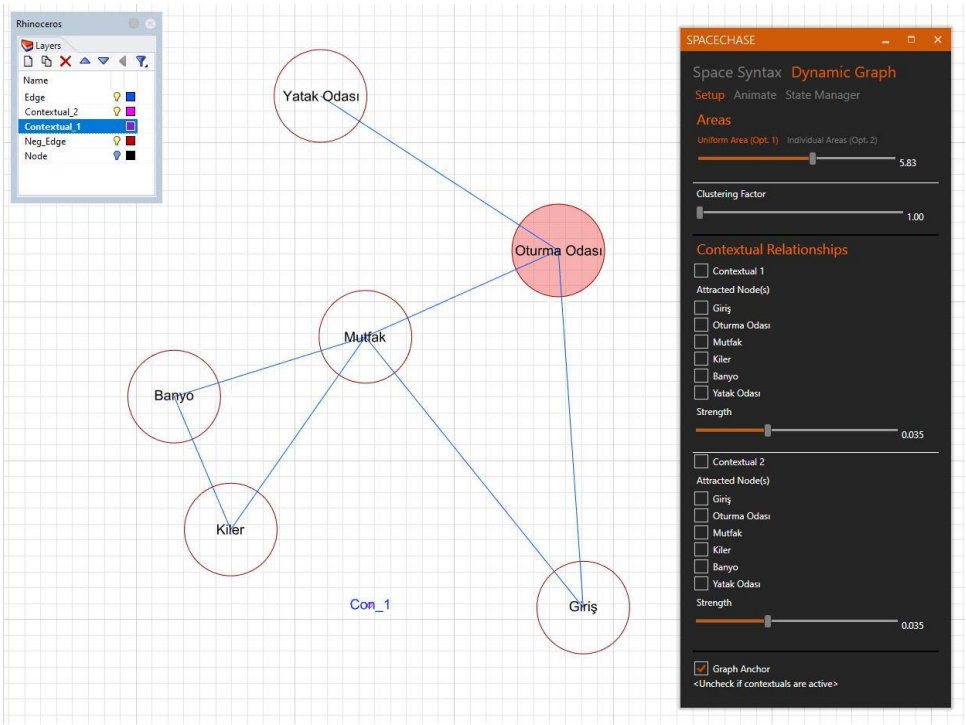


Şekil Ek 4-1. Giriş ekranı (*Space Syntax / Setup*). Düğüm noktalarının isimlendirildiği ekrandır. Kullanıcının Rhinoceros'un belirli isimlere sahip (*Node, Edge, Contextual_1*, vb.) katmanlarında çizdiği geometrik nesneler, SpaceChase tarafından otomatik olarak okunur ve bu ekranda isimlendirilir.

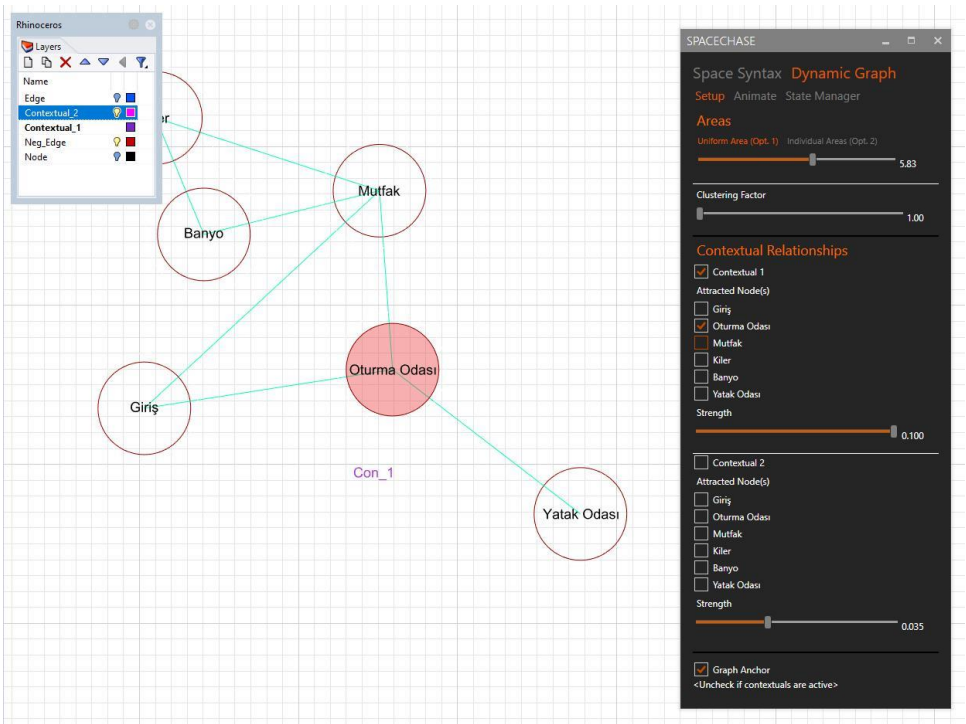


Şekil Ek 4-2. Analiz ekranı (*Space Syntax / Analyse*). Düğüm noktaları ve bağlantı bilgileri eklenti tarafından analiz edilirler. Düğüm noktalarının derinlik (*depth*) değerleri, ağın ortalama derinliği (*mean depth*) ve seçilen bir düğüm noktasının entegrasyon (*integration*) değerleri

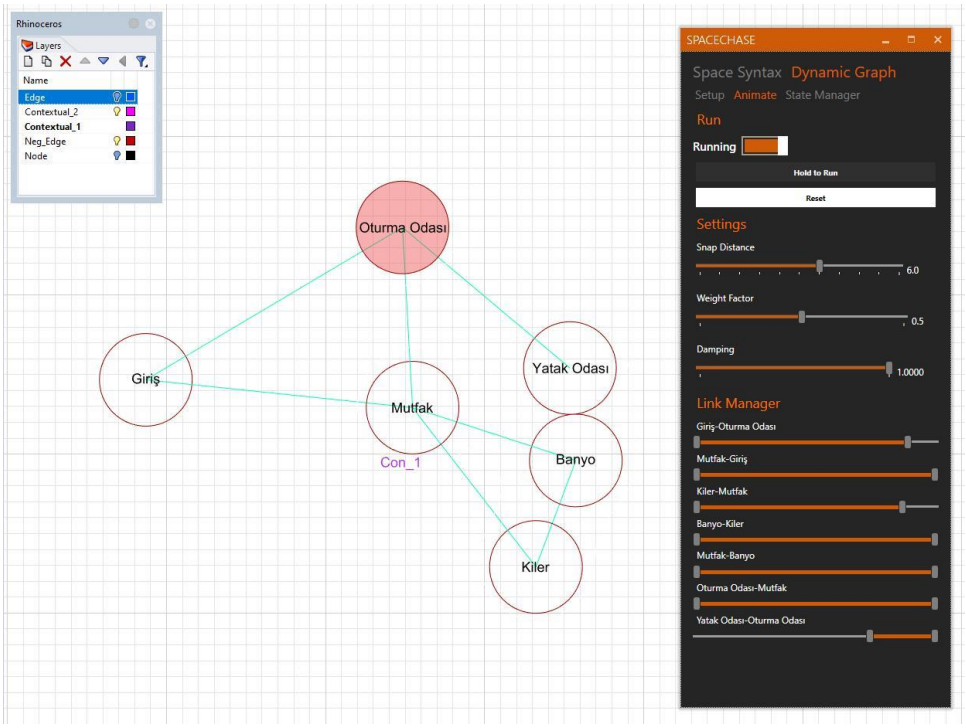
hesaplanarak bu ekranda görüntülenirler. Ayrıca seçilen düğüm noktasının geçiş grafiği (*justified graph*) oluşturularak bu ekranda görselleştirilir.



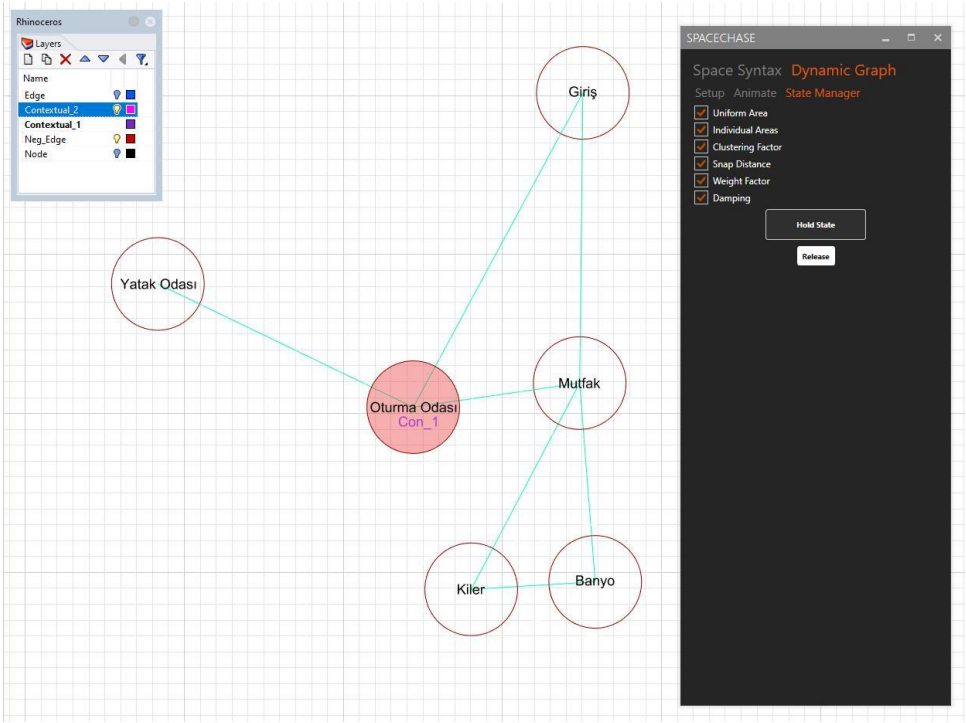
Şekil Ek 4-3. Dinamik ağ kurulumu (*Dynamic Graph / Setup*) Bu bölümde düğüm noktalarının boyutları, bağlamsal düğümler ve ağ animasyonunu kontrol eden diğer ayarlamalar yapılabilmektedir.



Şekil Ek 4-4. Bağlamsal düğüm noktalarının kullanımı (*Dynamic Graph / Setup / Contextual Relationships*) Bu prototip, kullanıcının iki adet noktayı bağlamsal düğüm olarak tanımlamasına izin vermektedir. Bu ekranda bağlamsal düğümler aktifleştirilerek hangi düğümleri hangi oranda çekecekleri ayarlanmaktadır.



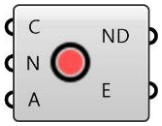
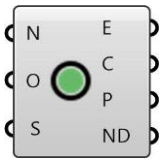
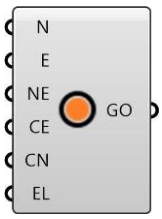
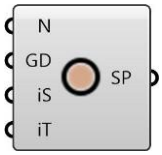
Şekil Ek 4-5. Dinamik ağ animasyonu (*Dynamic Graph / Animate*) "*Run*" düğmesi açılarak dinamik ağ görselleştirmesi başlatılır. Bu ekranda *Settings* bölümünde Kangaroo optimizasyon algoritmasına müdahale edilerek devinim biçimi kontrol edilebilir. Ayrıca *Link Manager* bölümünde ağ yapısındaki bağlantıların itme ve çekme etkileri kontrol edilebilir.

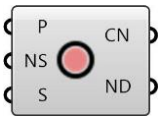
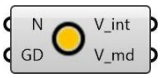
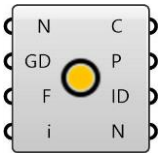
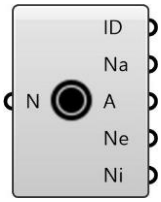
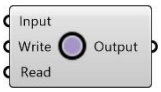


Şekil Ek 4-6. Durum yöneticisi ekranı (*Dynamic Graph / State Manager*) Bu ekran, geliştirilen arayüz kullanılarak yapılan değişikliklerin Grasshopper koduna aktarılmasını sağlamaktadır. Böylece yapılan ayarlamalar bir Grasshopper kodu olarak kaydedilebilir.

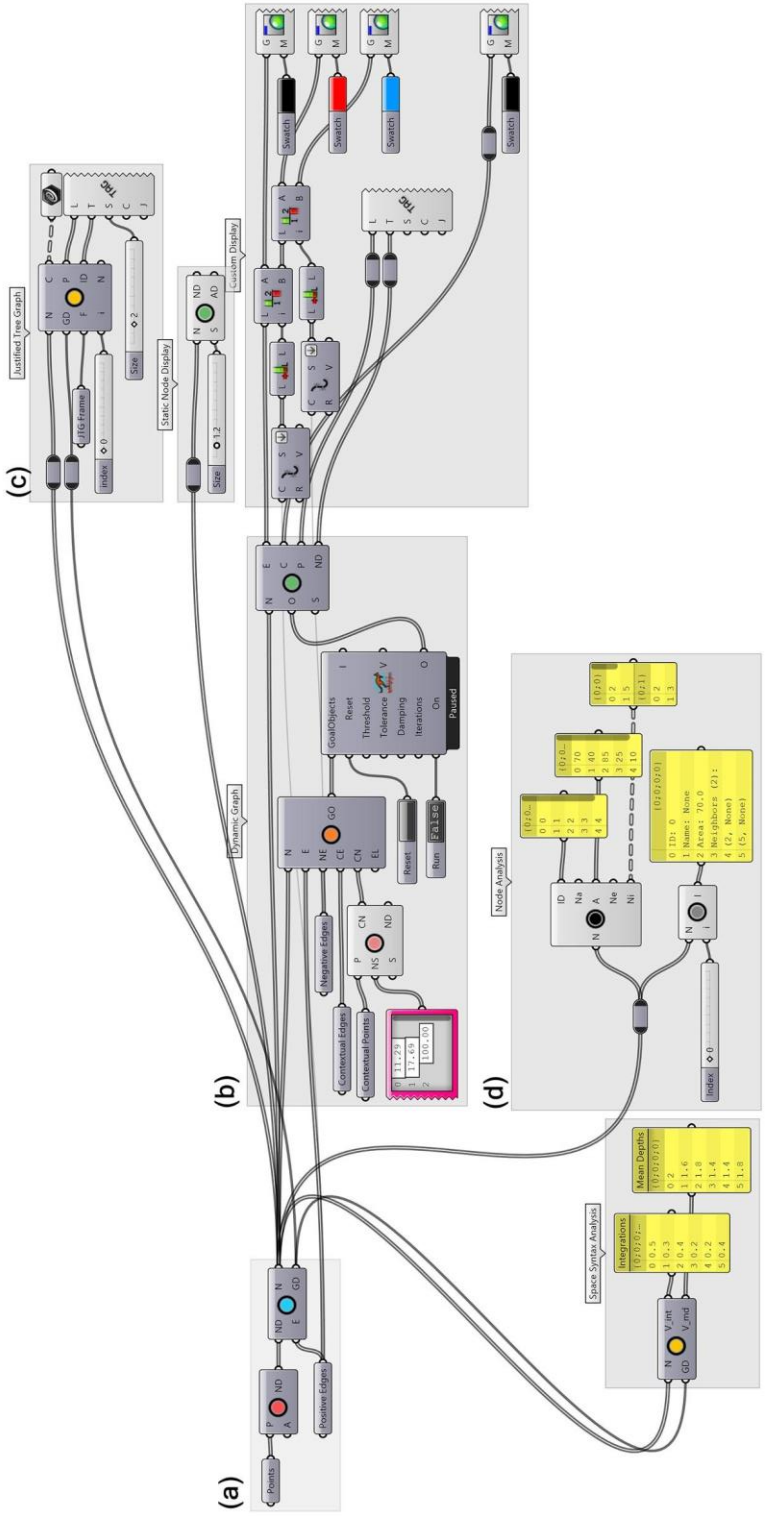
EK 5. SpaceChase for Grasshopper Beta (0.1.0) Geliştirilen Bileşenler

Tablo Ek 5-1. SpaceChase for Grasshopper 0.1.0 bileşenlerinin tanımları ve parametreleri.

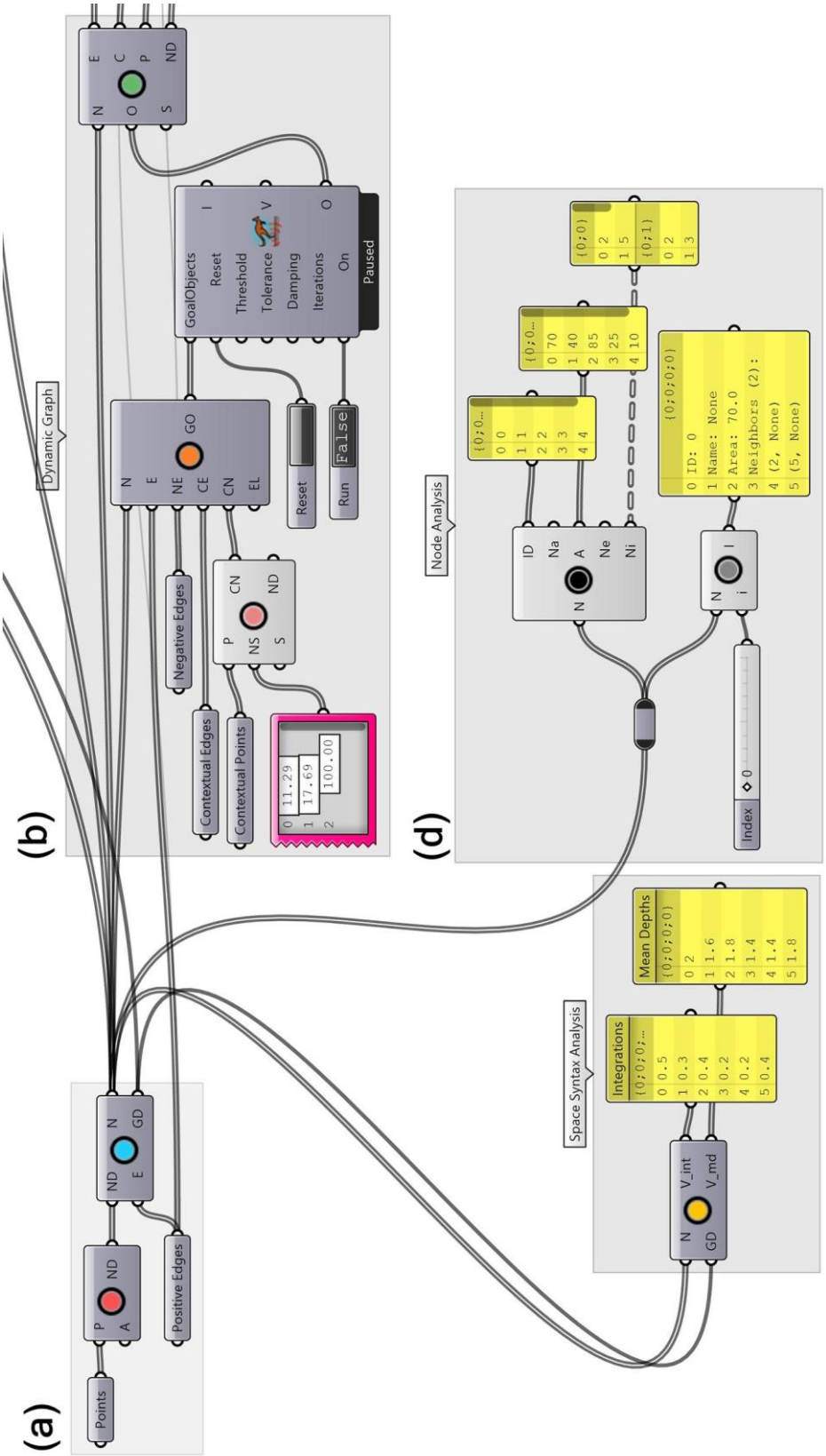
Bileşen	Tanım	Girdi(ler)	Çıktı(lar)
	**Construct Point Node:** Rhinoceros nokta objelerini kullanılarak SpaceChase düğüm noktası verisi üretir.	**P:** Rhinoceros nokta objelerinin ID'leri **A:** Düğüm noktalarının alanları	**ND:** SpaceChase düğüm noktası verileri
	**Construct Curve Node:** Rhinoceros çizgi objelerini kullanarak SpaceChase düğüm noktası verisi üretir.	**C:** Rhinoceros çizgi objelerinin ID'leri **N:** Düğüm noktalarının isimleri **A:** Düğüm noktalarının alanları	**ND:** SpaceChase düğüm noktası verileri **E:** SpaceChase ağ bağlantıları
	**Construct Graph:** SpaceChase ağ yapıları oluşturur.	**ND:** SpaceChase düğüm noktası verileri **E:** Rhinoceros çizgi objeleri (bağlantılar)	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ objeleri
	**SpaceChase Node Display:** SpaceChase düğüm noktası objelerinin ekranda görüntülenmesini sağlar.	**N:** SpaceChase düğüm noktası objeleri **S:** Yazı büyüklüğü	**ND:** Düğüm noktası isimleri **AD:** Düğüm noktası alanları
	**Dynamic Graph Display:** Kangaroo'da hesaplanan durumun görüntülenmesini sağlar.	**N:** SpaceChase düğüm noktası objeleri **O:** Kangaroo hedeflerinin hesaplanan sonuçları **S:** Yazı büyüklüğü	**E:** Ağ bağlantılarını gösteren çizgiler **C:** Düğüm noktalarını gösteren çemberler **P:** Düğüm noktalarını gösteren noktalar **ND:** Düğüm noktalarının isimleri
	**Construct Dynamic Graph:** SpaceChase ağ yapısının Kangaroo hedeflerine dönüştürülmesini sağlar. Böylece dinamik ve etkileşimli ağ yapısı kurulabilir.	**N:** SpaceChase düğüm noktası objeleri **E:** Rhinoceros çizgi objeleri (pozitif bağlantılar) **NE:** Rhinoceros çizgi objeleri (negatif bağlantılar) **CE:** Bağlamsal bağlantılar (opsiyonel) **CN:** Bağlamsal düğüm noktaları (opsiyonel) **EL:** Bağlantı uzunluğu listeleri (opsiyonel)	**GO:** Kangaroo hedefleri (*Goal Objects*)
	**Shortest Path:** Verilen iki düğüm noktası arasındaki en kısa mesafeyi hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ objesi **S:** Hesaplamanın başlayacağı düğüm noktasının indeks değeri	**SP:** En kısa yolu gösteren çizgi(ler)

		T: En kısa yoldan ulaşılması istenen düğüm noktasının indeks değeri	
	**Construct Contextual Nodes:** SpaceChase'de kullanılmak üzere bağlamsal düğüm noktaları üretir.	**P:** Nokta koordinatları **S:** Noktaların çekim etkisi (*strength*)	**CN:** SpaceChase bağlamsal düğüm noktası objeleri **ND:** Bağlamsal düğüm noktalarının isimleri
	**SS Integration:** SpaceChase düğüm noktalarının ortalama derinlik ve entegrasyon değerlerini hesaplar.	**N:** SpaceChase düğüm noktaları objeleri **GD:** SpaceChase ağ objesi	**V_int:** Entegrasyon değeri **V_md:** Ortalama derinlik değeri
	**Justified Tree Graph:** Mekan Dizimi analizlerinde kullanılan ağaç gösteriminin çizilmesini sağlar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ objesi **F:** Ağaç grafının içine çizileceği çerçeve çizgileri **i:** Ağaç grafının başlangıç noktası kabul edilecek düğüm noktasının indeks değeri	**C:** Graf bağlantılarını gösteren çizgiler **P:** Grafta düğüm noktalarını temsil eden noktalar **ID:** Düğüm noktalarının ID numaraları **N:** Düğüm noktalarının isimleri
	**Deconstruct Nodes:** SpaceChase düğüm noktası verilerini okunabilir ve kullanılabilir hale getirir.	**N:** SpaceChase düğüm noktası objeleri	**ID:** Düğüm noktasının ID numarası **Na:** Düğüm noktalarının isimleri **A:** Düğüm noktalarının alanları **Ne:** Her düğüm noktası için bağlantılı düğüm noktalarının isimleri **Ni:** Her düğüm noktası için bağlantılı düğüm noktalarının ID numaraları
	**Deconstruct Node:** Bir düğüm noktası hakkında bilgi verir.	**N:** SpaceChase düğüm noktası objeleri **i:** Seçilen düğüm noktası objesinin indeks değeri	**I:** Düğüm noktasının verileri (yazı formatında)
	**Pickler:** Girdi verilerini Rhino dosyası dizinine kaydeder (ileri çalışmalarda kullanılmak üzere geliştirilmiştir).	**Input:** Kaydedilecek veri **Write:** Kaydetme aktivasyonu **Read:** Kayıtlı veriyi okuma aktivasyonu	**Output:** Okunan veri

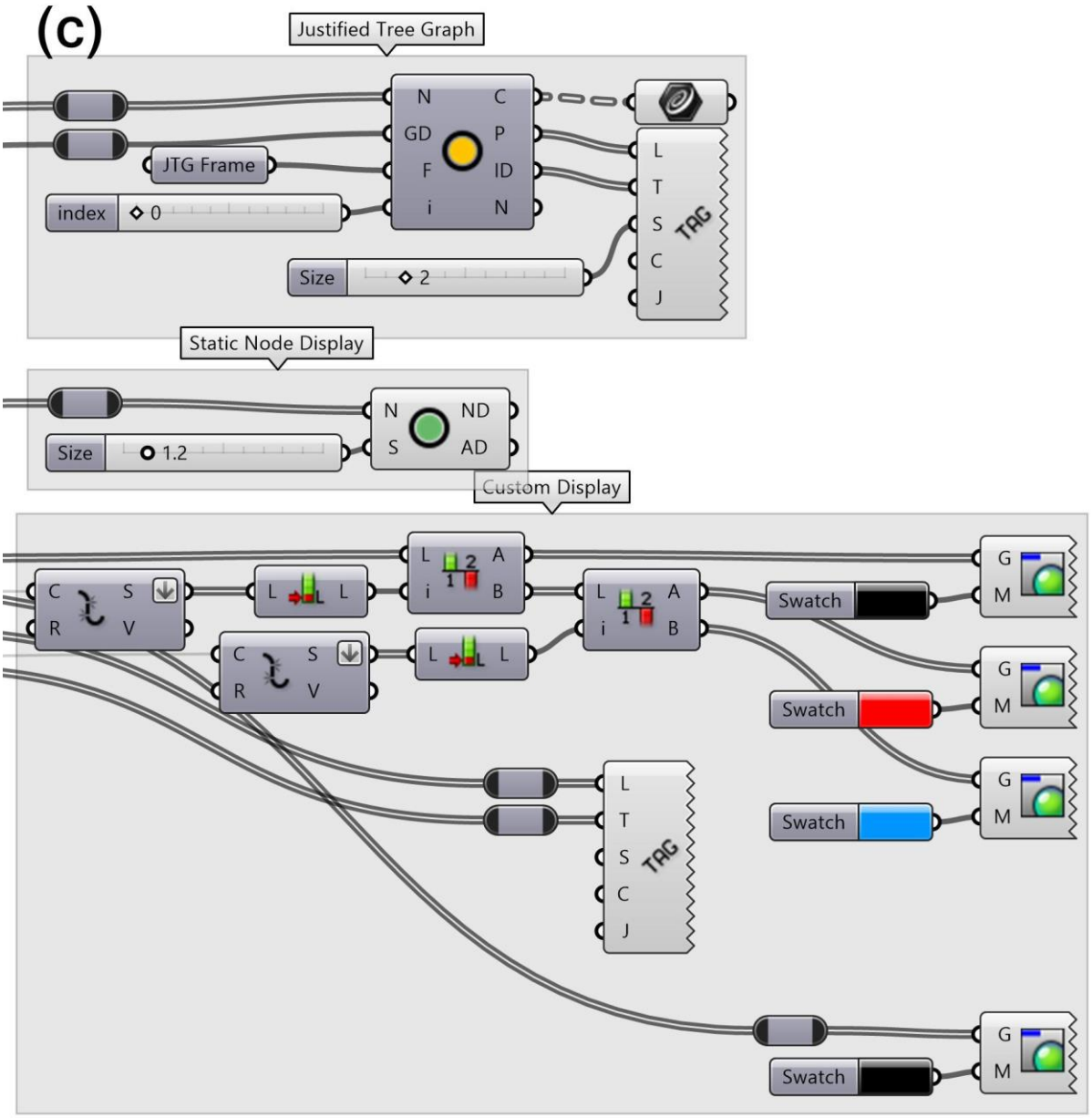
EK 6. SpaceChase for Grasshopper Beta (0.1.0) Örnek Veri akış Diyagramı



Şekil Ek 6-1. SpaceChase for Grasshopper Beta (0.1.0). Örnek veri akış diyagramı. (a), (b), (c), ve (d) gruplarının genel görünümü.



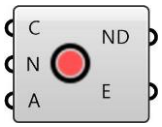
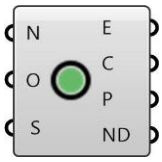
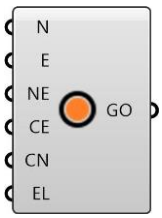
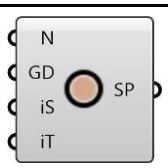
Şekil Ek 6-2. SpaceChase for Grasshopper Beta (0.1.0). Örnek veri akış diyagramı. (a), (b), ve (d) gruplarının detay görünümü.

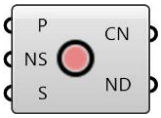
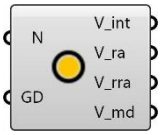
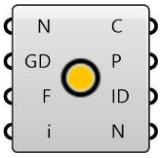
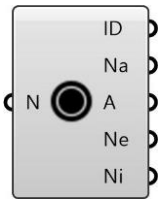


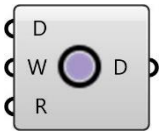
Şekil Ek 6-3. SpaceChase for Grasshopper Beta (0.1.0). Örnek veri akış diyagramı. (c) grubunun detay görünümü.

EK 7. SpaceChase for Grasshopper Beta (0.2.5) Geliştirilen Bileşenler

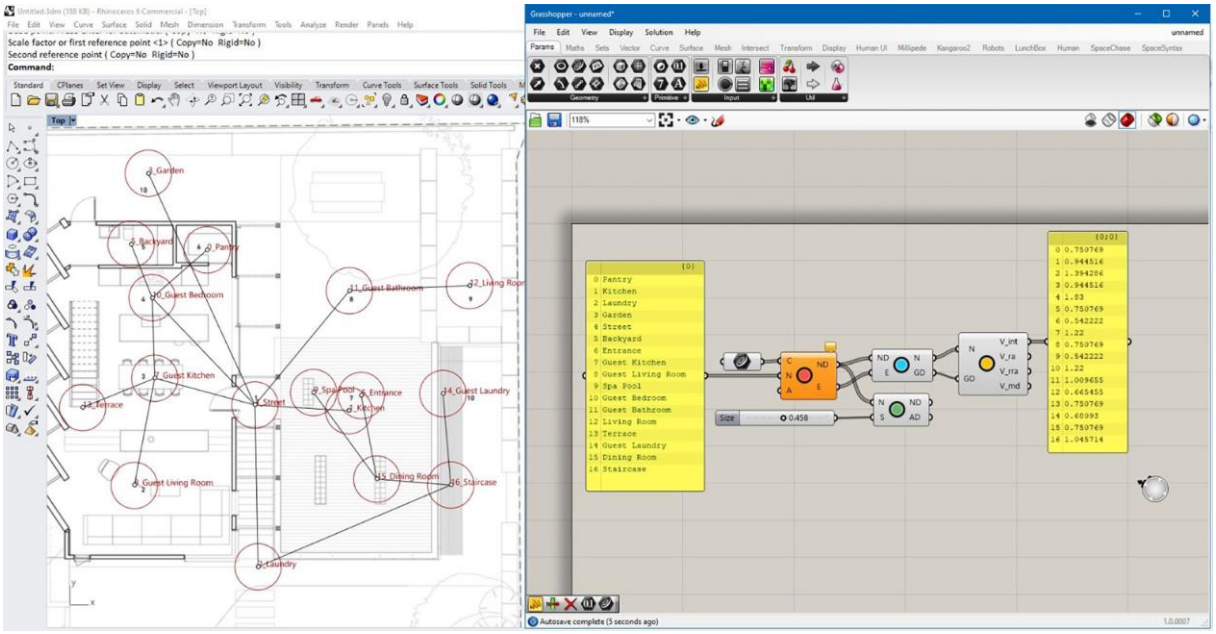
Tablo Ek 7-1. SpaceChase for Grasshopper 0.2.5 bileşenlerinin tanımları ve parametreleri.

Bileşen	Tanım	Girdi(ler)	Çıktı(lar)
	**Construct Point Node:** Rhinoceros nokta objelerini kullanılarak SpaceChase düğüm noktası verisi üretir.	**P:** Rhinoceros nokta objelerinin ID'leri **A:** Düğüm noktalarının alanları	**ND:** SpaceChase düğüm noktası verileri
	**Construct Curve Node:** Rhinoceros çizgi objelerini kullanarak SpaceChase düğüm noktası verisi üretir.	**C:** Rhinoceros çizgi objelerinin ID'leri **N:** Düğüm noktalarının isimleri **A:** Düğüm noktalarının alanları	**ND:** SpaceChase düğüm noktası verileri **E:** SpaceChase ağ bağlantıları
	**Construct Graph:** SpaceChase ağ yapıları oluşturur.	**ND:** SpaceChase düğüm noktası verileri **E:** Rhinoceros çizgi objeleri (bağlantılar)	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ objeleri
	**SpaceChase Node Display:** SpaceChase düğüm noktası objelerinin ekranda görüntülenmesini sağlar.	**N:** SpaceChase düğüm noktası objeleri **S:** Yazı büyüklüğü	**ND:** Düğüm noktası isimleri **AD:** Düğüm noktası alanları
	**Dynamic Graph Display:** Kangaroo'da hesaplanan durumun görüntülenmesini sağlar.	**N:** SpaceChase düğüm noktası objeleri **O:** Kangaroo hedeflerinin hesaplanan sonuçları **S:** Yazı büyüklüğü	**E:** Ağ bağlantılarını gösteren çizgiler **C:** Düğüm noktalarını gösteren çemberler **P:** Düğüm noktalarını gösteren noktalar **ND:** Düğüm noktalarının isimleri
	**Construct Dynamic Graph:** SpaceChase ağ yapısının Kangaroo hedeflerine dönüştürülmesini sağlar. Böylece dinamik ve etkileşimli ağ yapısı kurulabilir.	**N:** SpaceChase düğüm noktası objeleri **E:** Rhinoceros çizgi objeleri (pozitif bağlantılar) **NE:** Rhinoceros çizgi objeleri (negatif bağlantılar) **CE:** Bağlamsal bağlantılar (opsiyonel) **CN:** Bağlamsal düğüm noktaları (opsiyonel) **EL:** Bağlantı uzunluğu listeleri (opsiyonel)	**GO:** Kangaroo hedefleri (*Goal Objects*)
	**Shortest Path:** Verilen iki düğüm noktası arasındaki en kısa mesafeyi hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ objesi **S:** Hesaplamanın başlayacağı düğüm noktasının indeks değeri	**SP:** En kısa yolu gösteren çizgi(ler)

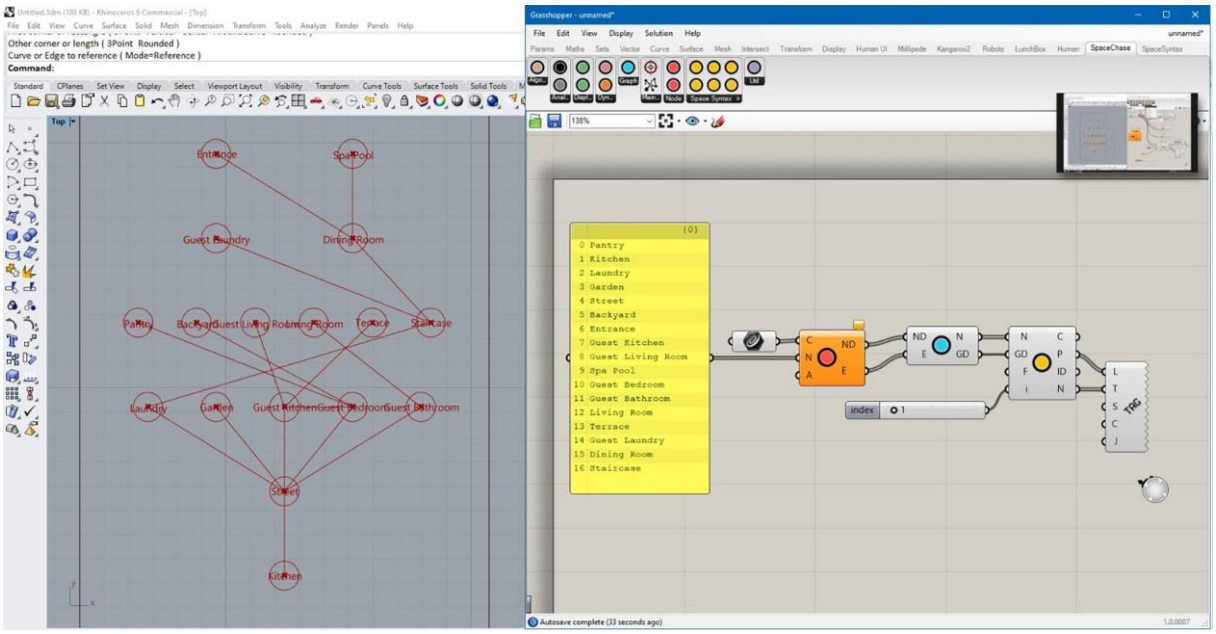
		T: En kısa yoldan ulaşılması istenen düğüm noktasının indeks değeri	
	**Construct Contextual Nodes:** SpaceChase'de kullanılmak üzere bağlamsal düğüm noktaları üretir.	**P:** Nokta koordinatları **S:** Noktaların çekim etkisi (*strength*)	**CN:** SpaceChase bağlamsal düğüm noktası objeleri **ND:** Bağlamsal düğüm noktalarının isimleri
	**Choice:** Mekan Dizimi "seçim" değerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı	**V_cho:** "Seçim" değeri
	**Choice EX:** Mekan Dizimi "seçim" değerini başlangıç ve bitiş adımlarını saymayarak hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı	**V_cho:** Başlangıç ve bitişler hariç tutulan "Seçim" değeri
	**Connectivity:** Mekan Dizimi "bağlılık" değerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri	**V_con:** "Bağlılık" değeri
	**Control:** Mekan Dizimi "kontrol" değerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri	**V_ctr:** "Kontrol" değeri
	**Integration:** Mekan Dizimi "ortalama derinlik", "entegrasyon", "relative asymmetry" ve "real relative asymmetry" değerlerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı	**V_int:** "Entegrasyon" değeri **V_ra:** "RA" değeri **V_rra:** "RRA" değeri **V_md:** "Ortalama Derinlik" değeri
	**Justified Tree Graph:** Mekan Dizimi analizlerinde kullanılan ağaç gösteriminin çizilmesini sağlar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ objesi **F:** Ağaç grafının içine çizileceği çerçeve çizgileri **i:** Ağaç grafının başlangıç noktası kabul edilecek düğüm noktasının indeks değeri	**C:** Graf bağlantılarını gösteren çizgiler **P:** Grafta düğüm noktalarını temsil eden noktalar **ID:** Düğüm noktalarının ID numaraları **N:** Düğüm noktalarının isimleri
	**Deconstruct Nodes:** SpaceChase düğüm noktası verilerini okunabilir ve kullanılabilir hale getirir.	**N:** SpaceChase düğüm noktası objeleri	**ID:** Düğüm noktasının ID numarası **Na:** Düğüm noktalarının isimleri **A:** Düğüm noktalarının alanları **Ne:** Her düğüm noktası için bağlantılı düğüm noktalarının isimleri **Ni:** Her düğüm noktası için bağlantılı düğüm noktalarının ID numaraları

	**Deconstruct Node:** Bir düğüm noktası hakkında bilgi verir.	**N:** SpaceChase düğüm noktası objeleri **i:** Seçilen düğüm noktası objesinin indeks değeri	**I:** Düğüm noktasının verileri (yazı formatında)
	**Pickler:** Grasshopper dosyasından bağımsız veri saklayabilmek için programlanan deneysel bileşen	**D:** Kaydedilecek veri **W:** Kaydet komutu **R:** Oku komutu	**D:** Okunan veri

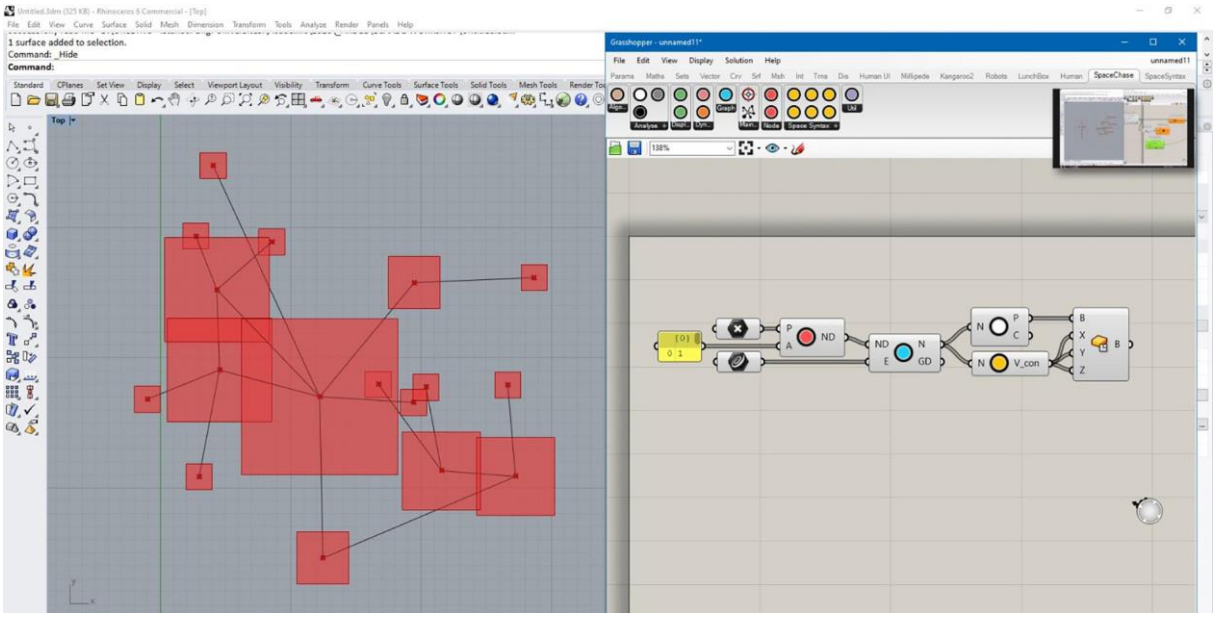
EK 8. SpaceChase for Grasshopper Beta (0.2.5) Örnek Veri akış Diyagramları



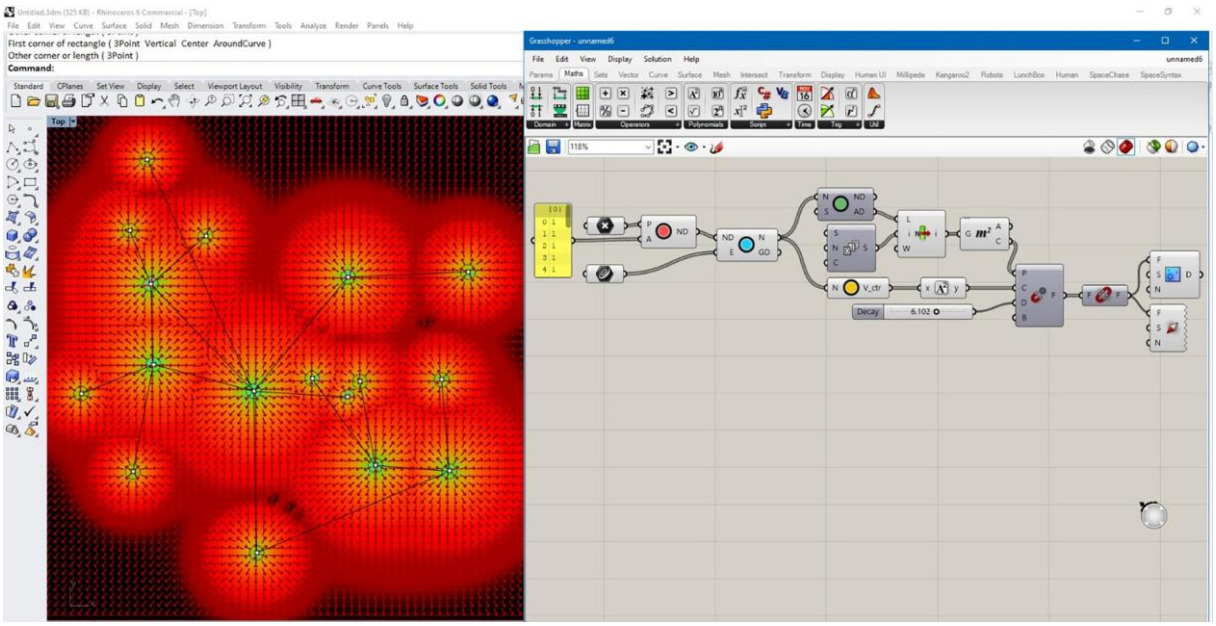
Şekil Ek 8-1. Bir plan çizimi üzerinde mekanları düğüm noktaları biçiminde tanımlanması. Bu örnekte SpaceChase, mekanların entegrasyon değerlerini hesaplamıştır.



Şekil Ek 8-2. Aynı örnek üzerinde, mekanların derinliklerini gösteren *Justified Tree Graph* oluşturulması.



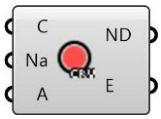
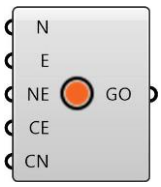
Şekil Ek 8-3. Bir başka görselleştirme çalışmasında Mekan Dizimi metriği olan kontrol, düğüm noktalarına yerleştirilen üç boyutlu küplerin boyutunu belirlemektedir. Bu tür çalışmalar Grasshopper'ın sunduğu olanaklarla zenginleştirilebilir.

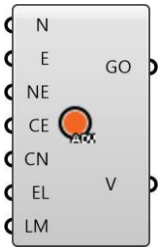
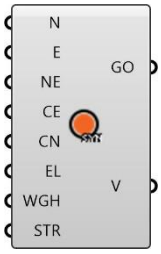
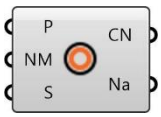
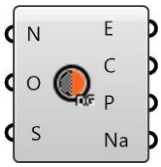


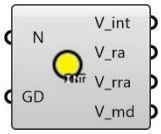
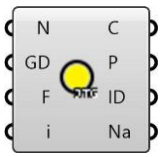
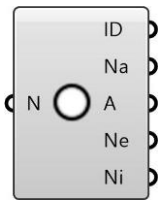
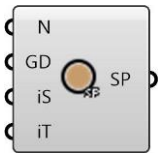
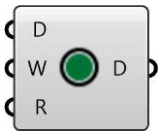
Şekil Ek 8-4. Düğüm noktalarını merkez alan bir vektör alanının (*vector field*) oluşturulması. Bu alanın her noktası için bir vektör tanımlanabilir ve bu vektörler düğüm noktalarına olan mesafelerine ve düğüm noktalarının çekim güçlerine bağlıdır. Bu örnekte düğüm noktalarının çekim gücü ise kontrol değerlerinden alınmaktadır.

EK 9. SpaceChase for Grasshopper Beta (0.3.3) Geliştirilen Bileşenler

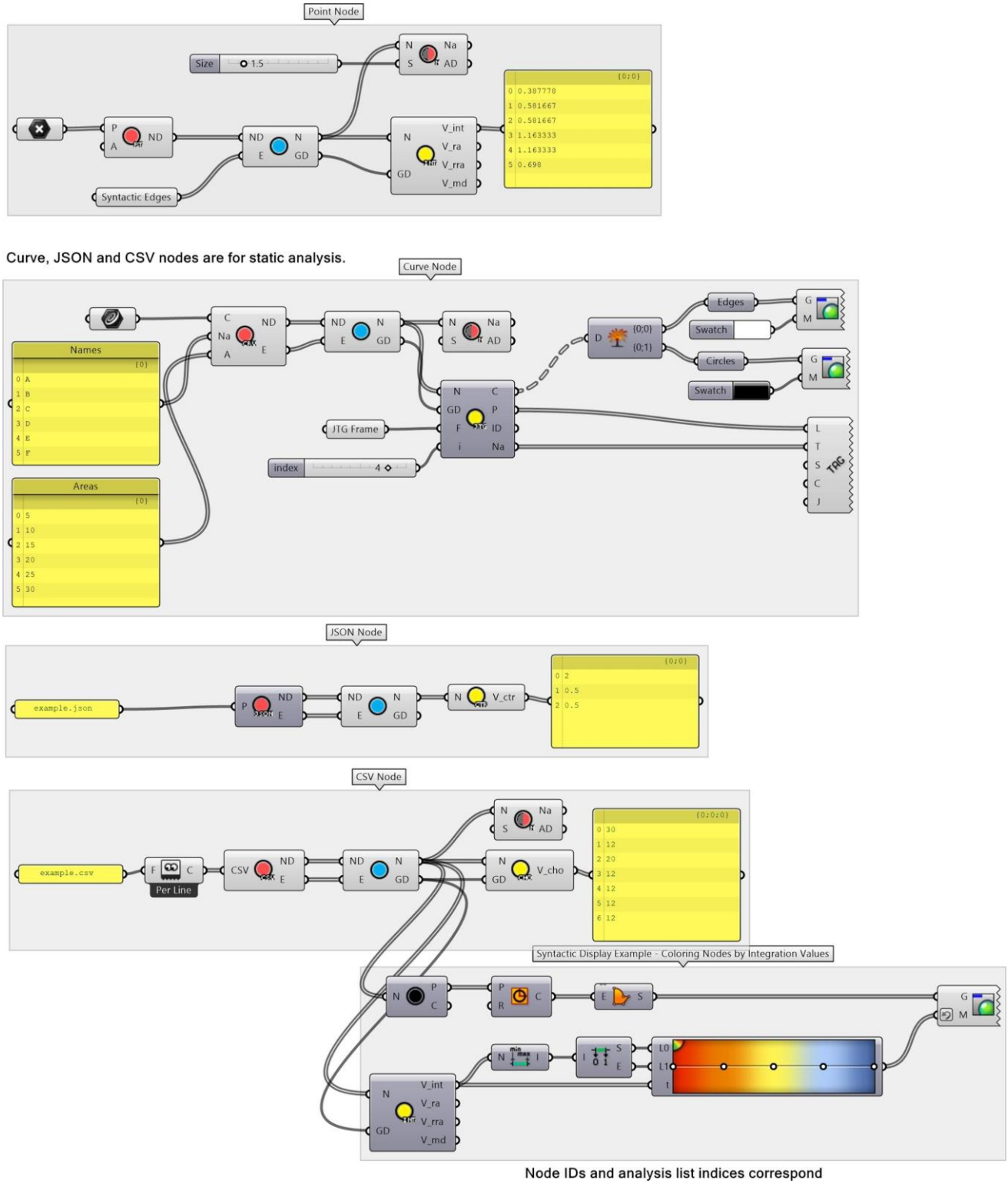
Tablo Ek 9-1. SpaceChase for Grasshopper 0.3.3 bileşenlerinin tanımları ve parametreleri.

Bileşen	Tanım	Girdi(ler)	Çıktı(lar)
	**Construct Point Node:** Rhinoceros nokta objelerini kullanarak SpaceChase düğüm noktası verisi üretir.	**P:** Rhinoceros nokta objeleri (opsiyonel olarak isim ve alan bilgisi barındıran) **A:** Düğüm noktalarının alanları (opsiyonel)	**ND:** SpaceChase düğüm noktası verileri
	**Construct Curve Node:** Rhinoceros çizgi objelerini kullanarak SpaceChase düğüm noktası ve bağlantı verisi üretir.	**C:** Rhinoceros çizgi objeleri **Na:** Düğüm noktalarının isimleri **A:** Düğüm noktalarının alanları	**ND:** SpaceChase düğüm noktası verileri **E:** SpaceChase ağ bağlantıları
	**Construct CSV Node:** CSV verisi kullanarak SpaceChase düğüm noktası ve bağlantı verisi üretir.	**CSV:** CSV dosya dizini	**ND:** SpaceChase düğüm noktası verileri **E:** SpaceChase ağ bağlantıları
	**Construct JSON Node:** SpaceChase oyunundan gelen JSON verisini SpaceChase'e aktarır.	**P:** JSON dosya dizini	**ND:** SpaceChase düğüm noktası verileri **E:** SpaceChase ağ bağlantıları
	**Node Display:** SpaceChase düğüm noktası objelerinin Rhinoceros ekranında görüntülenmelerini sağlar.	**N:** SpaceChase düğüm noktaları **S:** Yazı büyüklüğü	**Na:** Görünür yazı objeleri **AD:** Düğüm noktalarının alanlarını temsil eden çemberler
	**Construct Graph:** SpaceChase ağ yapısını oluşturur.	**ND:** SpaceChase düğüm noktası verileri **E:** SpaceChase ağ bağlantıları (çizgi objeleri)	**N:**SpaceChase düğüm noktası objeleri **GD:** Matematiksel ağ objesi
	**Construct Basic Dynamic Graph:** Basit dinamik ağ hesaplamaları yapar.	**N:** SpaceChase düğüm noktası objeleri **E:** SpaceChase ağ bağlantıları **NE:** Negatif SpaceChase ağ bağlantıları **CE:** (Pozitif) Bağlamsal SpaceChase ağ bağlantıları **CN:** Bağlamsal SpaceChase düğüm noktası objeleri	**GO:** Kangaroo hedefleri

	**Construct Advanced Dynamic Graph:** Detaylı kullanıcı insiyatifini mümkün kılan ileri düzey dinamik ağ hesaplamaları yapar ve arayüz pencereleri üretir.	**N:** SpaceChase düğüm noktası objeleri **E:** SpaceChase ağ bağlantıları **NE:** Negatif SpaceChase ağ bağlantıları **CE:** (Pozitif) Bağlamsal SpaceChase ağ bağlantıları **CN:** Bağlamsal SpaceChase düğüm noktası objeleri **EL:** Bağlantı uzunluklarının eşitlenmesi (opsiyonel) **LM:** *Link Manage*r arayüzü aktivatörü	**GO:** Kangaroo hedefleri **V:** Dinamik ağ taşıyan vektör
	**Construct Syntactic Dynamic Graph:** Mekan Dizimi metriklerinin hesaba katılacağı dinamik ağ hesaplamaları yapar.	**N:** SpaceChase düğüm noktası objeleri **E:** SpaceChase ağ bağlantıları **NE:** Negatif SpaceChase ağ bağlantıları **CE:** (Pozitif) Bağlamsal SpaceChase ağ bağlantıları **CN:** Bağlamsal SpaceChase düğüm noktası objeleri **EL:** Bağlantı uzunluklarının eşitlenmesi **WGH:** "Ağırlık" Mekan Dizimi metrikleri listesi **STR:** "Güç" Mekan Dizimi metrikleri listesi	**GO:** Kangaroo hedefleri **V:** Dinamik ağ taşıyan vektör
	**Construct Contextual Node:** Bağlamsal SpaceChase düğüm noktası verisi üretir.	**P:** Rhinoceros nokta objeleri **NM:** Çekim etkisinin ayarlanacağı *Contextual Node Manager* arayüzünü açan aktivatör **S:** Bağlamsal düğüm noktası isminin yazı boyutu	**CN:** SpaceChase bağlamsal düğüm noktası objeleri **Na:** Düğüm noktalarının isimlerini göstern yazı objeleri
	**Dynamic Graph Display:** Kangaroo'da hesaplanan dinamik ağın Rhinoceros ekranında görüntülenmesini sağlar.	**N:** SpaceChase düğüm noktası objeleri **O:** Kangaroo hedeflerinin "*GoalFunction*" çıktısı **S:** Yazı büyüklüğü	**E:** Dinamik ağın bağlantı çizgileri **C:** Dinamik ağın düğüm noktalarının alan çemberleri **P:** Dinamik ağın düğüm noktaları **Na:** Yazı objeleri
	**Choice:** Mekan Dizimi "seçim" değerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı	**V_cho:** "Seçim" değeri
	**Choice EX:** Mekan Dizimi "seçim" değerini başlangıç ve bitiş adımlarını saymayarak hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı	**V_cho:** Başlangıç ve bitişler hariç tutulan "Seçim" değeri
	**Connectivity:** Mekan Dizimi "bağlılık" değerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri	**V_con:** "Bağlılık" değeri
	**Control:** Mekan Dizimi "kontrol" değerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri	**V_ctr:** "Kontrol" değeri

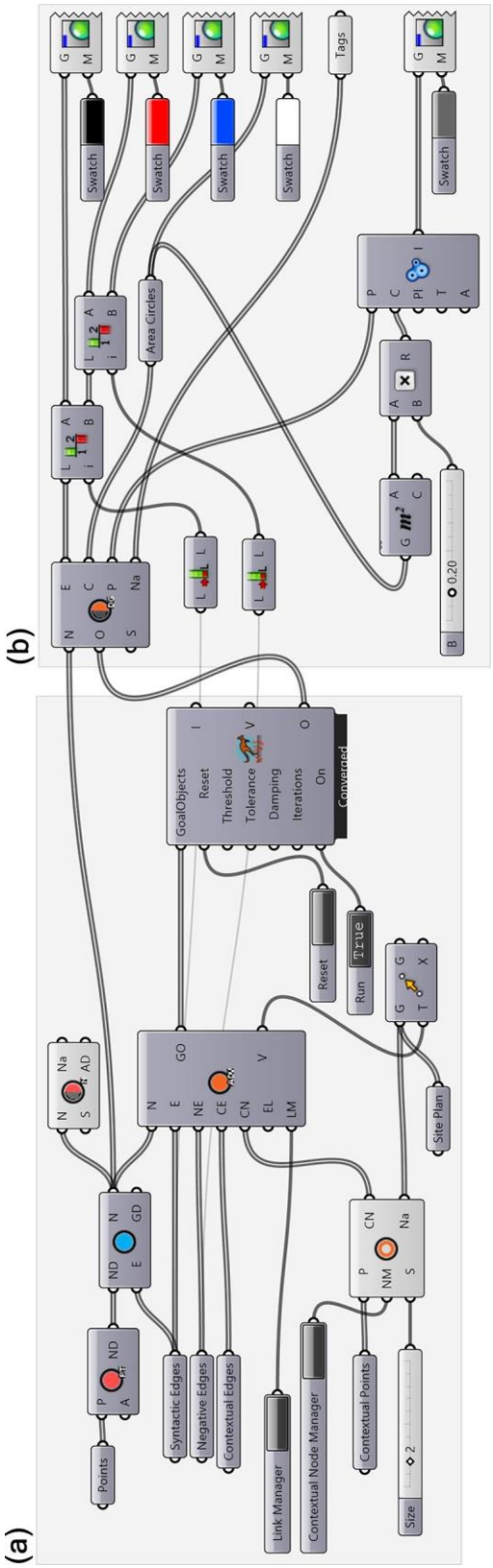
	**Integration:** Mekan Dizimi "ortalama derinlik", "entegrasyon", "relative asymmetry" ve "real relative asymmetry" değerlerini hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı	**V_int:** "Entegrasyon" değeri **V_ra:** "RA" değeri **V_rra:** "RRA" değeri **V_md:** "Ortalama Derinlik" değeri
	**Justified Tree Graph:** Mekan Dizimi analizlerinde kullanılan ağaç diyagramını çizer.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı **F:** Ağaç gösteriminin içine çizileceği çerçeve çizgileri **i:** Ağaç gösteriminin başlangıç noktası kabul edilecek düğüm noktasının indeks numarası	**C:** Diyagramın bağlantı çizgileri **P:** Diyagramın noktaları **ID:** Noktaların numaraları **Na:** Noktaların isimleri
	**Deconstruct Node:** SpaceChase düğüm noktası objelerini kullanılabilir nokta ve çizgi objelerine dönüştürür.	**N:** SpaceChase düğüm noktası objeleri	**P:** Nokta objeleri **C:** Çizgi objeleri
	**Node Analysis:** SpaceChase düğüm noktası objelerinin kullanılabilir verilerini listeler olarak sunar.	**N:** SpaceChase düğüm noktası objeleri	**ID:** Düğüm noktalarının indeksleri **Na:** Düğüm noktalarının isimleri **A:** Düğüm noktalarının alanları **Ne:** Bağlantıların isimleri **Ni:** Bağlantıların indeksleri
	**Node Information:** SpaceChase düğüm noktası objelerini okunabilir veri olarak sunar.	**N:** SpaceChase düğüm noktası objeleri **i:** Seçilen düğüm noktasının indeks değeri	**I:** Düğüm noktası bilgilerini içeren metini
	**Shortest Path:** Verilan iki düğüm noktası arasındaki en kısa mesafeyi hesaplar.	**N:** SpaceChase düğüm noktası objeleri **GD:** SpaceChase ağ yapısı **iS:** Başlangıç düğüm noktasının indeksi **İT:** Hedef düğüm noktasının indeksi	**SP:** En kısa yol (çizgi objeleri)
	**Pickler:** Grasshopper dosyasından bağımsız veri saklayabilmek için programlanan deneysel bileşen	**D:** Kaydedilecek veri **W:** Kaydet komutu **R:** Oku komutu	**D:** Okunan veri

EK 10. SpaceChase for Grasshopper Beta (0.3.3) Örnek Veri Akış Diyagramları

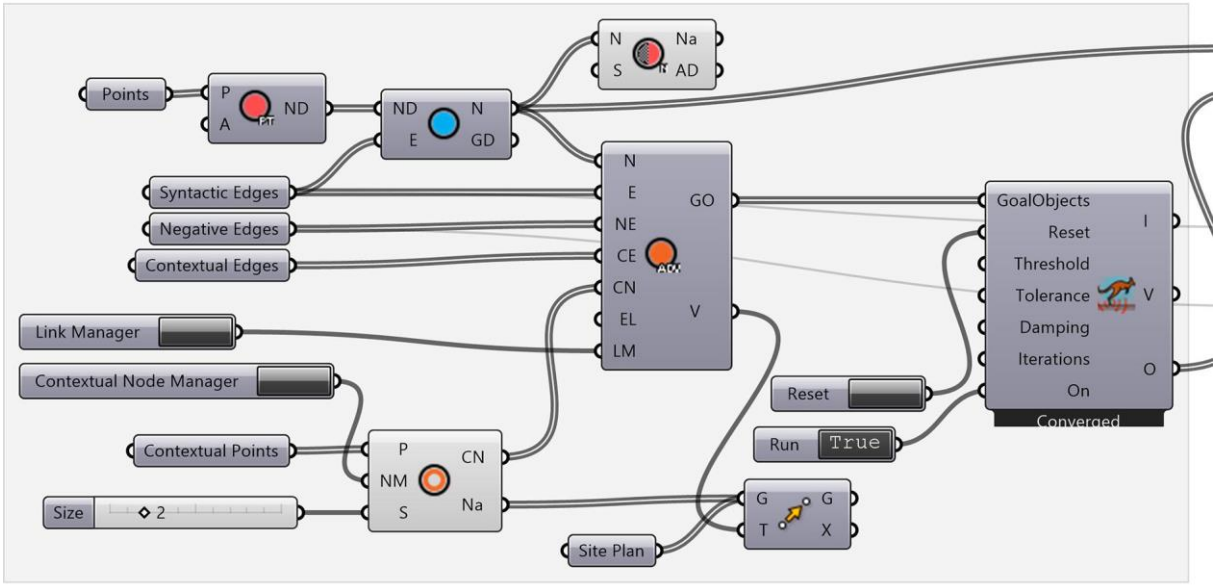


Şekil Ek 10-1. SpaceChase for Grasshopper 0.3.3 durağan ağ yapısı örnekleri. Farklı şekillerde düğüm noktası üreten bileşenlerin kullanım şekilleri görülmektedir.

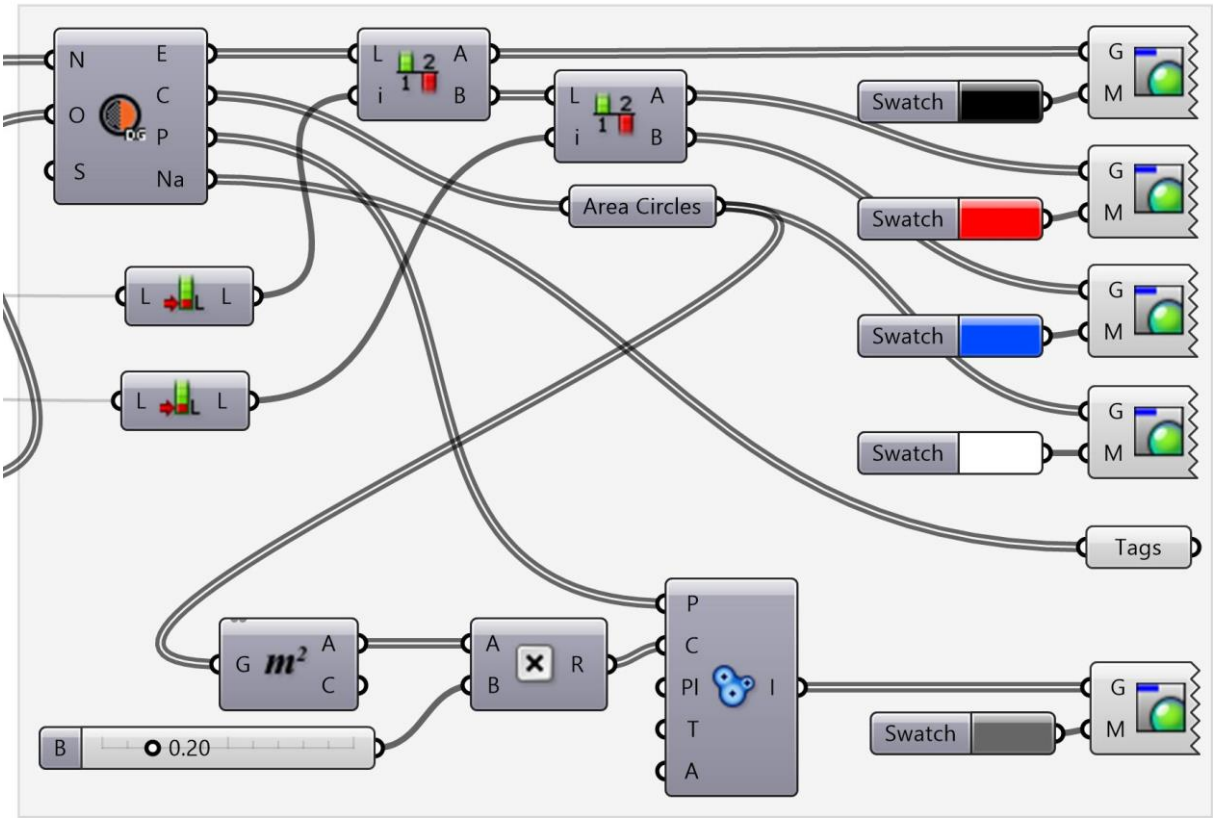
SpaceChase for Grasshopper Beta (0.3.3) Dinamik Ağ Yapısı Örneği



Şekil Ek 10-2. SpaceChase for Grasshopper 0.3.3 dinamik ağ yapısı için örnek veri akış diyagramı. (a) ve (b) gruplarının genel görünümü.



Şekil Ek 10-3. SpaceChase for Grasshopper 0.3.3 dinamik ağ yapısı için örnek veri akış diyagramı. (a) grubunun detay görünümü.



Şekil Ek 10-4. SpaceChase for Grasshopper 0.3.3 dinamik ağ yapısı için örnek veri akış diyagramı. (b) grubunun detay görünümü.

EK 11. SpaceChase for Grasshopper Değişim Günlüğü (*Changelog*)

Tablo Ek 11-1. Değişim günlüğü. Tarihleri, sürümleri ve yapılan değişiklikleri gösterir.

Date & version	**Changes**		
21.08.2020 v0.1.0	Includes:	Nodes:	Construct Point Node: Constructs nodes from Rhino points. Acquires names from point objects, requires gh input areas.
			Construct Curve Node: Constructs nodes from curve discontinuities. Requires gh input names and areas.
		Graph (Graph Theory):	Construct Graph: Constructs graph from nodes and edges. Calculates neighbor data, outputs graph nodes with graph information.
		Space Syntax:	SS Integration: Calculates Space Syntax integration and mean depth values from graph data
			Justified Tree Graph: Draws “Justified Tree Graph” inside given frame from node index.
		Dynamic Graph:	Construct Dynamic Graph: Prepares Kangaroo objects from nodes, can receive “Negative Edges”, “Contextual Edges” and “Contextual Nodes”. “Edge Lengths” input equalizes kangaroo clamps, default clamp behaviour takes edge lengths, which converts the graph into a weighted graph model in dynamic environment. Includes “Grab” and “Show” for interaction and display. Outputs kangaroo goal objects.
			Construct Contextual Node: Constructs contextual nodes from points and strengths (as float). Strengths affect connected contextual edge clamp lengths. Includes built-in node display.
		Analysis:	Deconstruct Nodes: Deconstructs node list into node properties as lists.
			Deconstruct Single Node: Deconstructs a single node with given index, outputs info as a more readable string format.
		Algorithms:	Shortest Path: Calculates unweighted shortest path between given nodes using Dijkstra’s algorithm, outputs shortest path curves from input edges.
		Display:	Node Display: Displays node ids, names and areas, uses TextGoo and circles.
			Dynamic Graph Display: Separates edges and points from GoalFunction outputs, provides names and areas in point order.
		Util:	Pickler: Experimental data serialization and deserialization utilizing Python Pickle module. Can be implemented into a data collection method.
22.08.2020 v0.1.1	Changes:	Justified Tree Graph:	Added “Names” output corresponding node names in points output list order.
24.08.2020 v0.2.0	Changes:	Node classes:	Added self.control under Node class constructor (__init__) to make the “Control” calculations possible
		SS Integration:	Renamed to “Integration”
		Pickler:	Renamed I/O with initials.
		Module imports:	Changed importing style
	Additions:	Space Syntax:	Control: Calculates and outputs space syntax control values.

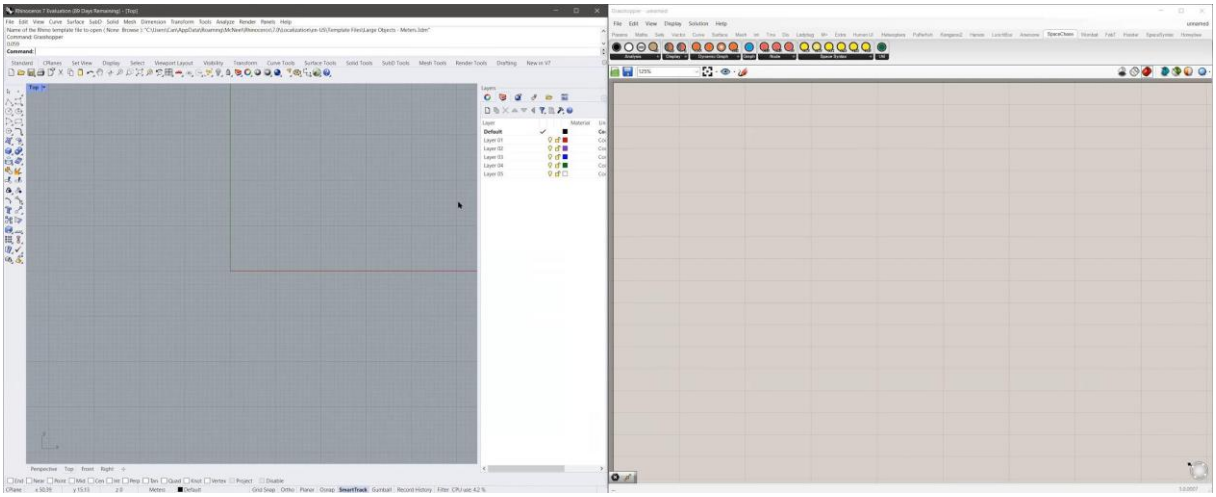
			Choice: Calculates and outputs space syntax choice values. Due to duplicate paths not being sorted (A to B / B to A), choice values may be double.
			Connectivity: Calculates and outputs neighbor counts (i.e. Connectivity)
25.08.2020 v0.2.1	Changes:	Choice:	Adjusted choice calculation to include origin and target nodes.
25.08.2020 v0.2.2	Changes:	Choice:	Adjusted choice calculation to include duplicate paths.
	Additions:	Space Syntax:	Choice EX: A second choice calculation component, excluding duplicate paths and origin/target nodes (same as v0.2.1 Choice).
25.08.2020 v0.2.3	Bug Fixes:	Construct Dynamic Graph:	Removed line: Con_PLine_List = list(dict.fromkeys(Con_PLines))
			Unsorted type (dictionary) was causing contextual point strengths order to shuffle.
25.08.2020 v0.2.4	Changes:	Integration:	Renamed old “Integration” values to “Relative Asymmetry”
	Additions:	Integration:	Added “Real Relative Asymmetry” calculation and output, along with DValues.py which consists of the 300 Dk values for RRA.
			Added “Integration” calculation and output, calculated from RRA (1/RRA).
	Corrections:	Pickler:	Corrected name/nickname order in code
14.09.2020 v0.2.5	Changes:	Deconstruct Nodes:	Renamed to “Node Analysis”
		Deconstruct Single Node:	Renamed to “Single Node Analysis”
	Additions:	Deconstruct Nodes:	Outputs node positions as list and edges as tree.
18.11.2020 v0.3.0	Changes:	Warning Messages:	Reduced less-than-important warning messages to remark level
		Component Classes:	Renamed all classes for code hygiene
		Analyse (Tab):	Renamed to “Analysis”
		Algorithm (Tab):	Removed tab
		Construct Dynamic Graph:	Renamed to “Construct Basic Dynamic Graph"
			Removed “Edge Length”
		Shortest Path:	Moved to Analysis
			Changed description
			Changed ID
		Deconstruct Nodes:	Renamed to “Deconstruct Node”
		Single Node Analysis:	Renamed to “Node Information”
			Changed description
		Dynamic Graph Display:	Changed description
		Construct Point Node:	Added support for single area value
			Added support for area input from point object properties

			Works in the form of “{Name} * {Area}” inside “Name” field
		Construct Curve Node:	Added support for single area value
		Component Icons:	Remade all icons, added text identifiers for similar components
		Dynamic Graph:	Fixed Translation Vector (Small unnoticed mistake, was causing slight misalignment)
	Additions:	Warning Messages:	Added commented-out Windows Forms pop-up for beta/trial expiration.
		Construct Dynamic Graph (now Advanced):	Added Translation Vector Output
		Construct Syntactic Dynamic Graph:	An alternative dynamic graph model, edge goals rely on given syntactic attributes as list.
		Construct Advanced Dynamic Graph:	Advanced dynamic graph model, relies on edge lengths, has optional inputs for uniform edge lengths and the “Link Manager” GUI.
		Link Manager:	Link Manager GUI for Advanced Dynamic Graph.
			Pop-up GUI initiated
			Push & Pull Sliders generated from edge count
			Slider names matched with corresponding nodes
			Slider values generate real-time values (forms.Form)
			Sliders keep values on GUI restart
		Construct Contextual Node:	Added Node Manager, a variant of link manager class, to replace gene pool usage.
		Nodes:	Construct CSV Node: Constructs node data from given .cvs input, utilises CurveNode node class.
19.11.2020 v0.3.1	Bug Fixes:	Point Node Class:	Fixed class constructor for the cases where there’s no name input
			Fixed class constructor for float area input
21.11.2020 v0.3.2	Additions:	Construct JSON Node:	Constructs nodes and edges from json file format, to be used as a medium for data transmission between SpaceChase game and plugin.
	Changes:	Point Node Class:	Redefined class constructor for every case regarding name and area input presency.
		Curve Node Class:	Fixed class constructor to create meaningful position data
24.11.2020 v.3.3	Bug Fixes:	Node Manager Class:	Fixed a bug involving refreshing Node Manager on changes in node count
		Link Manager Class:	Fixed a bug involving refreshing Link Manager on changes in link count

EK 12. SpaceChase for Grasshopper Temel Kullanım Kılavuzu

1. Kurulum

SpaceChase for Grasshopper'ın çalıştırılabilmesi için Windows tabanlı bilgisayarınızda Rhinoceros 6 (SR25 veya sonrası) yüklü olmalıdır. Eklenti Rhinoceros 7 altında da çalışabilmektedir. Eklentinin son sürümü olan 0.3.3 (10.01.2021 itibariyle), http://spacechase.app web adresinden indirilebilmektedir. İndirilen zip klasörü içerisinde çeşitli örnek çalışmaların yanında SpaceChase033.ghpy isimli bir dosya bulunmaktadır. Bilgisayarınızda Rhinoceros 6 yazılımı çalıştırıldıktan ve Grasshopper komutu ile Grasshopper arayüzü açıldıktan sonra .ghpy uzantılı dosyayı fare yardımıyla sürükleyerek Grasshopper kanvasının (boş pencerenin) üzerine bırakarak yüklemeyi tamamlayabilirsiniz. Yükleme sorunsuz tamamlandıysa Grasshopper penceresinde en üst satırdaki komut sekmelerinde *SpaceChase* isimli bir sekme belirmelidir (Şekil Ek 12-1).

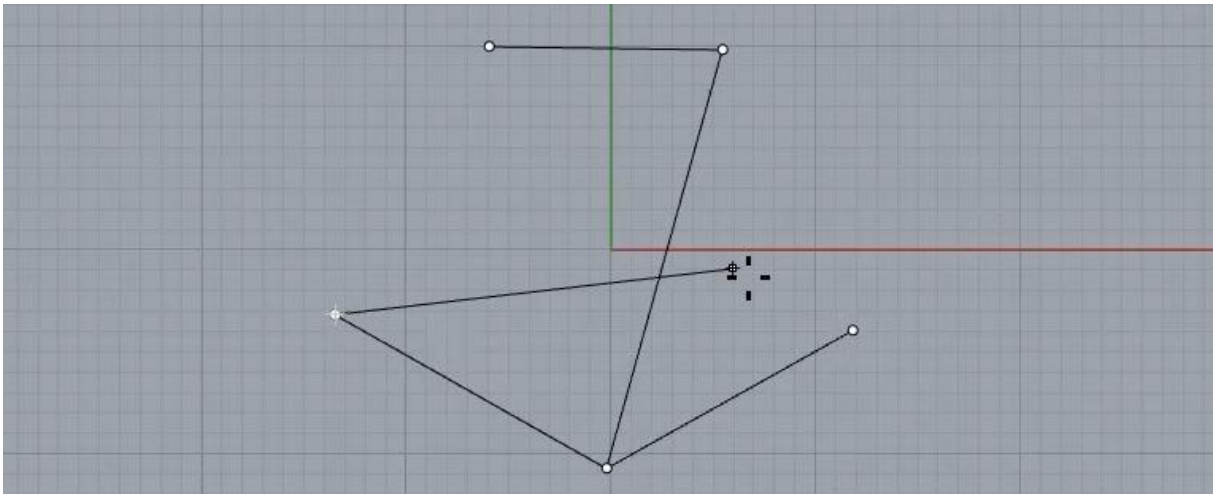


Şekil Ek 12-1. Rhinoceros ve Grasshopper üzerinde SpaceChase ile çalışma ortamı. Solda, Rhinoceros ekranı, ortada katmanlar (*Layer*), sağda ise Grasshopper kanvası ve SpaceChase bileşenleri görülmektedir.

2. Temel Ağ Yapısının Tanımlanması

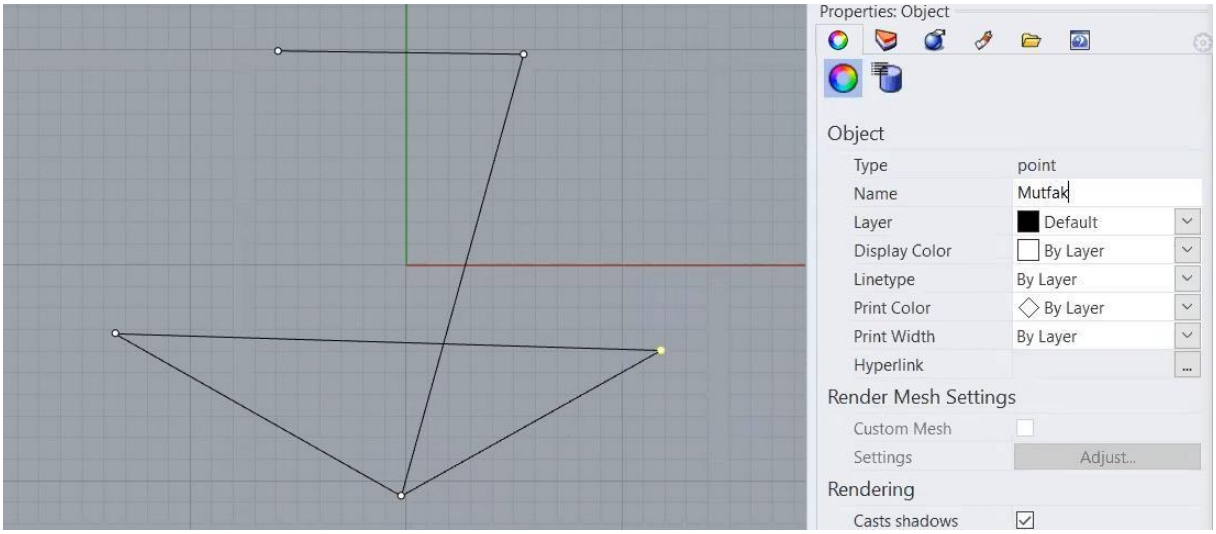
SpaceChase ile ağ yapılarını tanımlarken, mekansal bilgileri içeren düğüm noktaları (*Node*), ve düğüm noktaları arasındaki ilişkileri ve bağlantıları temsil eden bağlantılar (*Edge*) kullanılır. SpaceChase for Grasshopper ile çalışmaya başlamadan önce başlangıç ağ yapısını Rhinoceros üzerinde tanımlamak gerekmektedir. Rhinoceros'un nokta (*Point*) nesneleri kullanılarak mekanları veya işlevleri temsil eden düğüm noktaları oluşturulabilir, çizgi komutu kullanılarak bağlantılar oluşturulabilir. Şekil Ek 12-2'de, Rhinoceros ekranında yerleştirilen bir

dizi nokta ve bu noktaları birleştiren çizgiler görülmektedir. Çizgilerin noktaları hassas biçimde yakalayabilmesi için Rhinoceros'un *Object Snap (Osnap) Point* seçeneği aktif olmalıdır.



Şekil Ek 12-2. Rhinoceros üzerinde düğüm noktalarını *point* komutu ile, bağlantıları ise *line* veya *polyline* komutu ile tanımlamak mümkündür.

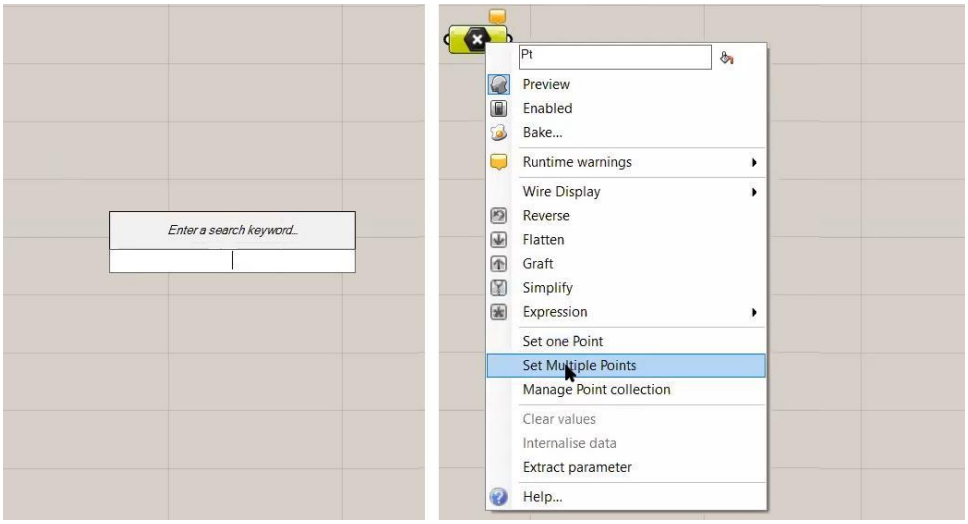
Temel ağ elemanlarının tanımlanmasının ardından düğüm noktalarına isim verilebilir. Bunun için, bir nokta nesnesini seçip Rhinoceros'un obje özellikleri (*Object Properties*) penceresinde bulunan isim (*Name*) kutusuna giriş yapılabilir (Şekil Ek 12-3). Aşağıdaki örnekte bazı mekan isimlerinin noktalara girilmesi gösterilmiştir.



Şekil Ek 12-3. Düğüm noktalarının Rhinoceros içerisinde isimlendirilmesi.

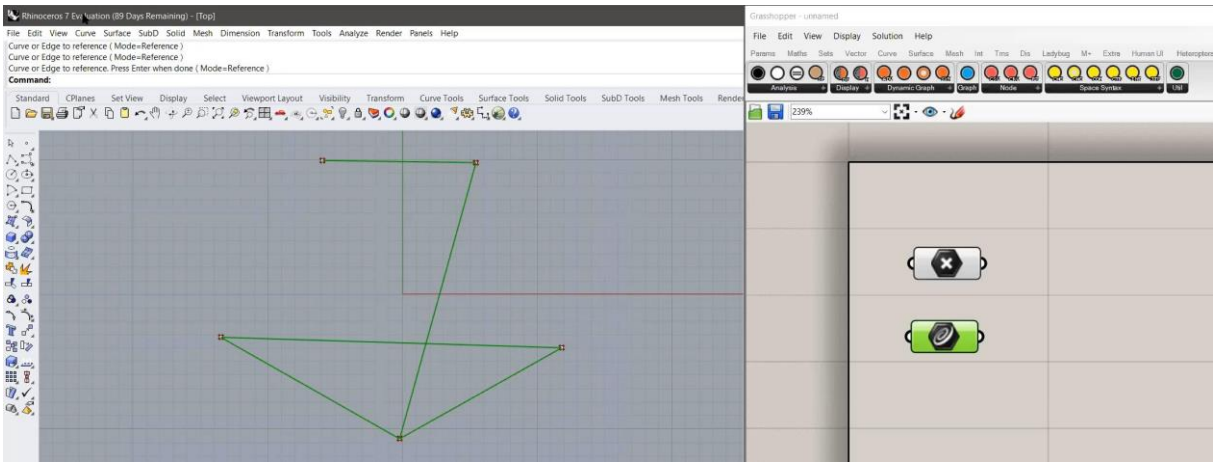
SpaceChase işlevlerini kullanabilmek için Rhinoceros'ta yukarıdaki gibi tanımlanan ve isimlendirilen düğüm noktaları Grasshopper içerisine yüklenmelidir. Bunun için Grasshopper kanvasına bir *Point Parameter* bileşeni eklenir (Şekil Ek 12-4-sol). Bu bileşen boş bir nokta

konteynırıdır. Bu bileşene sağ tıklanarak *Set Multiple Points* seçeneği ile Rhinoceros'taki noktaların Grasshopper ortamına aktarımı tamamlanır (Şekil 4-sağ).



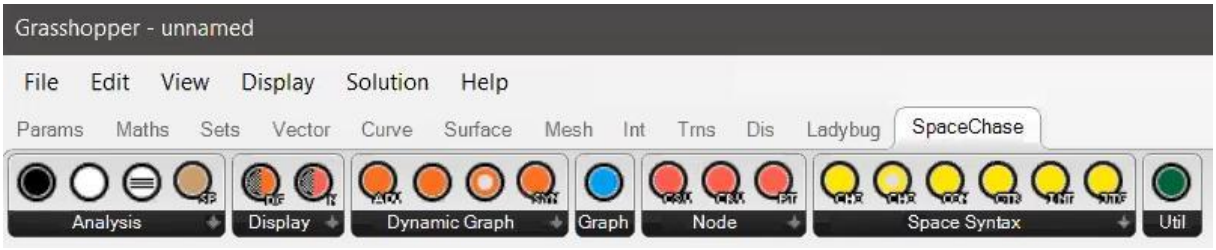
Şekil Ek 12-4. Sol; Grasshopper kanvasına çift tıklamak suretiyle açılan arama (*search*) kutucuğu. Bu kutucuğa *point parameter* yazılarak bulunan bileşen kanvasa yerleştirilir. Sağ; Kanvasa yerleştirilen bileşene sağ tıklandığında açılan menüden seçilen *set multiple points* seçeneği.

Aynı yöntem kullanılarak bağlantıların bilgisini içeren çizgilerin de Grasshopper'a aktarılması gerekmektedir. Bu defa *Curve Parameter* bileşeni yerleştirilmeli ve ekrandaki çizgiler *Set Multiple Curves* seçeneği ile Grasshopper içerisine aktarılmalıdır. Bu iki bileşenin sağ tarafında bulunan çıkıntılar kullanılarak bu verilerin başka bileşenlerin sol tarafındaki girdilere bağlanması sağlanacaktır. Bu işlem sonucunda Şekil Ek 12-5'teki ekran görüntüsü elde edilecektir.

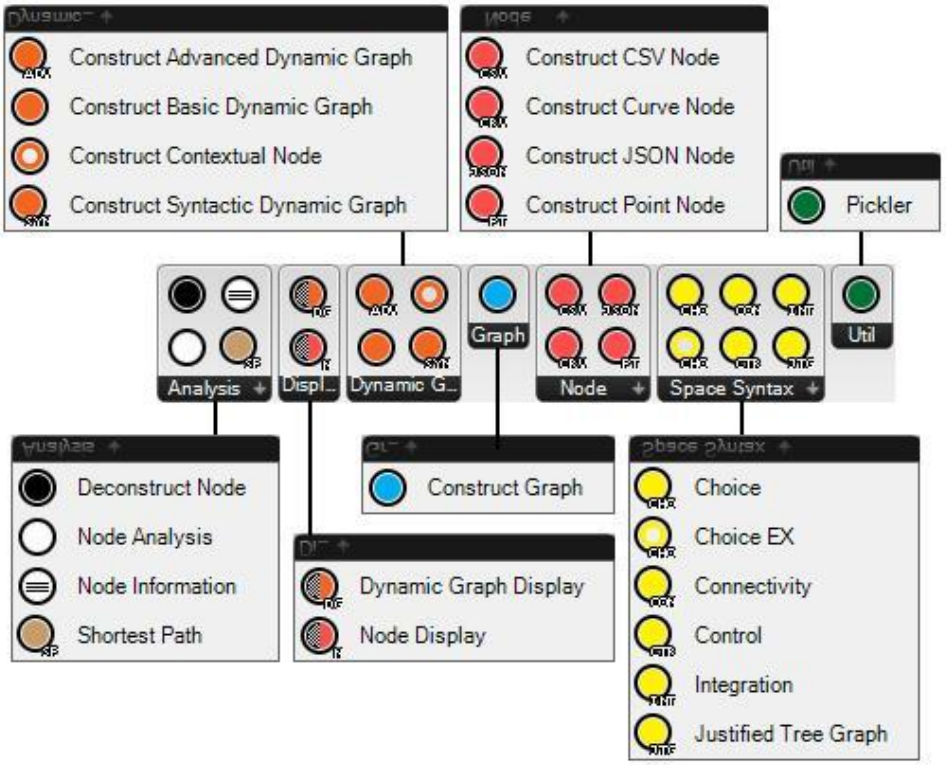


Şekil Ek 12-5. Rhinoceros'ta çizilen bağlantıların *curve parameter* bileşenin içerisine aktarılmasından sonraki ekran görüntüsü.

Rhinoceros üzerinde düğüm noktalarının ve bağlantıların tanımlanarak Grasshopper içerisine aktarılmasının ardından SpaceChase'in kullanımına geçilebilir. SpaceChase, Şekil Ek 12-6 ve 7'de görülen bileşenlere sahiptir.



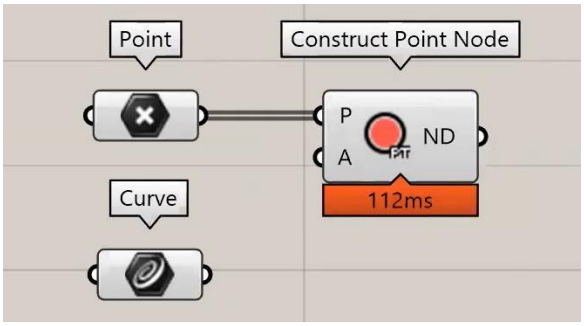
Şekil Ek 12-6. SpaceChase sekmesinde bulunan bileşenler 7 grup içerisinde organize edilmiştir.



Şekil Ek 12-7. SpaceChase bileşen gruplarının içerisinde bulunan bileşenler.

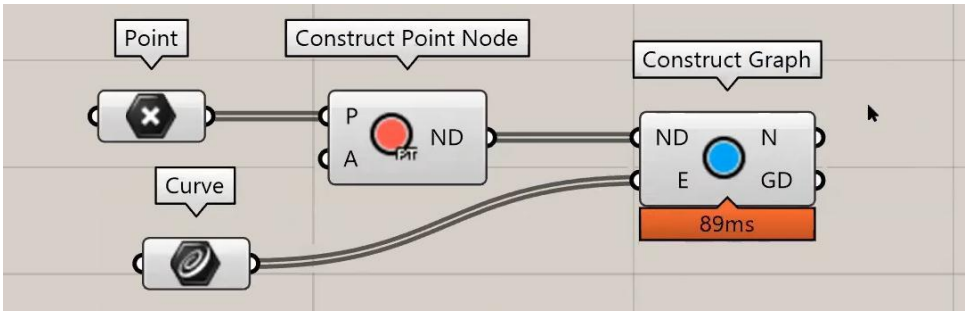
Az önce yerleştirilen *Point Parameter* ve *Curve Parameter* bileşenlerinin çıktıları olan nokta ve çizgilerin SpaceChase'in okuyup işleyebileceği ağ yapılarına dönüştürülmeleri gerekmektedir. SpaceChase'in düğüm noktaları *Node* (N) olarak isimlendirilmiştir. Düğüm noktalarını oluşturan verilerin (koordinat, isim, ağırlık vb.) toplamı da *Node Data* (ND) veri tipi olarak tanımlanmıştır. Rhinoceros'ta tanımlanan nokta verisini SpaceChase'in okuyup işleyebileceği düğüm noktalarının verisine (ND) dönüştürebilmek için SpaceChase sekmesindeki *Node* başlıklı bileşen grubu alındaki *Point Node* bileşeni kullanılmalıdır. Bu grupta bulunan diğer

bileşenler (*CSV Node, Curve Node*) farklı türdeki girdilerden *Node Data* türetmek için alternatif yöntemler sunmaktadırlar. Bu bileşenlerin açıklamaları Ek 9'da sunulmuştur. Point Node bileşeni kanvasa yerleştirildikten sonra P girdisi, az önce oluşturulan *Point Parameter*'in çıktısına bağlanmalıdır (Şekil Ek 12-8). *Point Node* bileşeninin sağ tarafındaki çıktı olan (ND) *Node Data* üretmektedir. Bu bileşenin A girdisi opsiyoneldir ve istendiğinde düğüm noktalarının alan bilgisinin girilebilmesini sağlar.



Şekil Ek 12-8. *Point Node* bileşeninin *Point Parameter* ile bağlanarak *Node Data* (ND) çıktısını türetmesi.

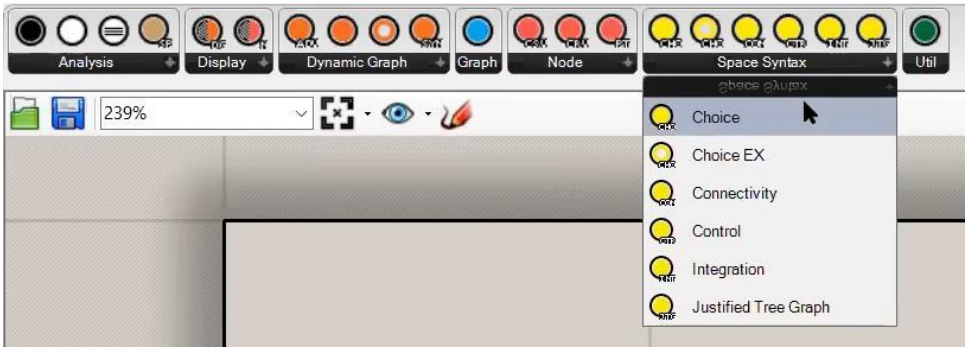
SpaceChase sekmesinde Graph isimli bileşen grubu altında bulunan *Construct Graph* bileşeni, düğüm noktalarını ve bağlantıları, SpaceChase'in okuyabileceği ve işleyebileceği ağ yapılarına dönüştürmektedir. Bu bileşeni kanvasa yerleştirip az önce üretilen *Node Data*'nın ND girdisine, çizgilerin de E girdisine bağlanması gerekmektedir (Şekil Ek 12-9). Bu bileşenin çıktıları olan N ve GD, SpaceChase'in pek çok işlevinde kullanılan temel ağ verileridir. N, SpaceChase düğüm noktalarını, GD ise SpaceChase ağ yapısını ifade etmektedir.



Şekil Ek 12-9. *Construct Graph* bileşeni ile düğüm (ND) ve bağlantı (E) verilerinden SpaceChase düğüm (N) ve ağ (GD) verilerinin türetilmesi.

3. Mekan Dizimi Analizleri

Oluşturulan ağ yapısının Mekan Dizimi metriklerinin hesaplanması mümkündür. Bunun için SpaceChase sekmesindeki *Space Syntax* başlıklı bileşen grubunda bulunan bileşenler kullanılmaktadır (Şekil Ek 12-10).



Şekil Ek 12-10. SpaceChase'in Mekan Dizimi analizlerini yapan bileşenleri.

Bu bileşenlerin işlevleri aşağıda özetlenmiştir ve her bileşenin Grasshopper girdi ve çıktı parametreleri Ek 9'da açıklanmıştır.

Choice: Mekan Dizimi'nin seçim değerini hesaplayan bileşendir.

Choice EX: Mekan Dizimi'nin seçim değerini başlangıç ve bitiş adımlarını saymayarak hesaplar.

Connectivity: Mekan Dizimi'nin bağlılık değerini hesaplar.

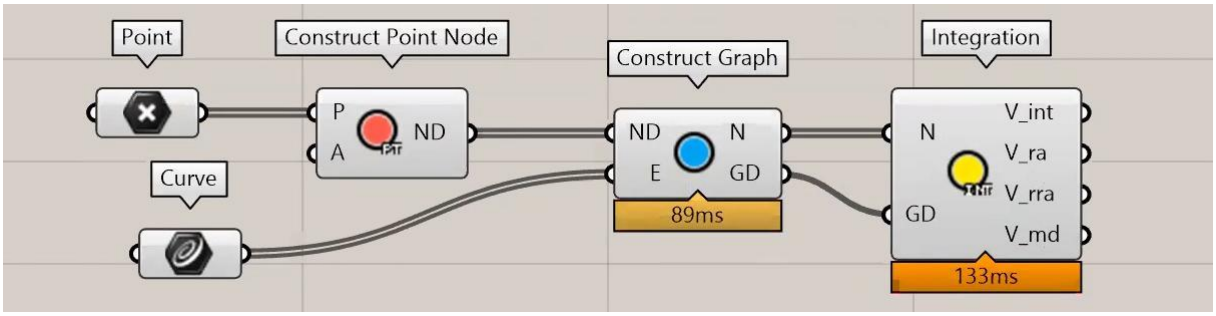
Control: Mekan Dizimi'nin kontrol değerini hesaplar.

Integration: Mekan Dizimi'nin ortalama derinlik, entegrasyon (*relative assymetry* ve *real relative assymetry)* değerlerini hesaplar.

Justified Tree Graph: Mekan Dizimi analizlerinde kullanılan ilişkisel derinlik haritalarını çizer.

Bu analizlerden kısaca söz etmek gerekirse; Mekan Dizimi metrikleri içinde "bağlılık" her mekandan bir adım uzaklıkta olan mekan sayısını bildirir. Derinlik ise başlangıç olarak kabul edilen bir mekandan ulaşılmak istenen mekana giderken atılması gereken adım sayısıdır (Hillier ve Hanson, 1984). Mekan Dizimi teknikleri bağlamında mekansal ilişkileri temsil eden geçiş grafiği içinde bir mekandan diğerine direkt olarak geçilebiliyorsa o mekanın derinliği 1, bir ara mekan geçilmesi gerekiyorsa 2 olarak kabul edilmektedir. "Ortalama derinlik" ise geçiş grafiğinde her derinlikteki yer alan mekan sayısının toplamının tüm sistemdeki mekanların bir eksiğine bölünmesiyle elde edilen değerdir. Bir mekanın "entegrasyon" değeri grafik içindeki tüm diğer mekanlardan o mekanın rölatif derinliğidir (Hillier ve Hanson, 1984). Bir başka deyişle entegrasyon, o mekandan çizilen ilişki grafiğinin biçiminin sayısal bir ifadesini oluşturur. Bir mekan ne kadar entegre ise sistemdeki diğer tüm mekanları yakınına çekecek, ne kadar ayrı ise diğer tüm mekanları kendisinden uzaklaştıracaktır. Entegre mekanlar harekette

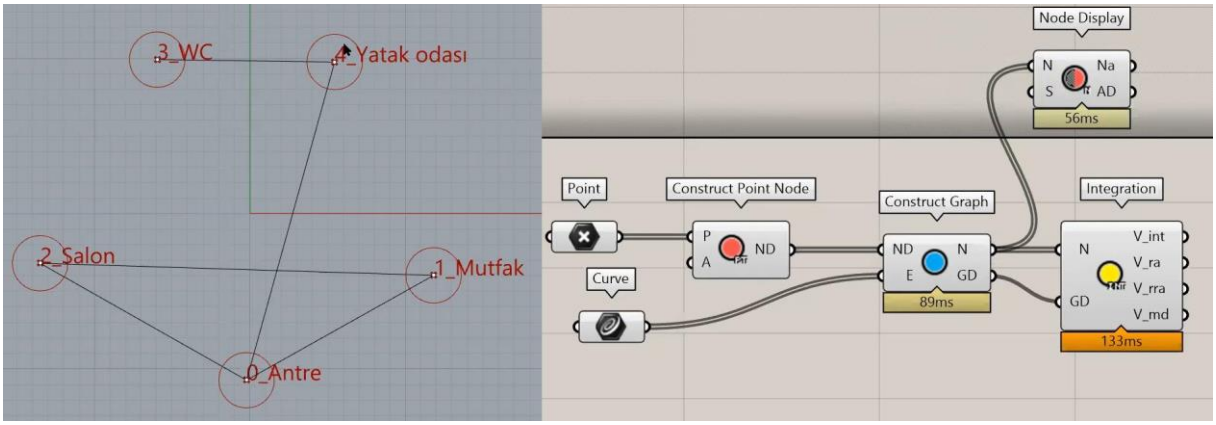
zengin, ayrılmış mekanlar harekette fakir mekanlar olarak düşünülebilir. “Kontrol”, hareket edilebilecek bir mekan olarak her bir mekanın yakın komşuları için ne derece seçenek sunduğunu ölçer (Hillier ve Hanson, 1984). Mekanın kontrol değeri hesaplanırken her mekanın belirli sayıda, “n”, yakın komşusu olduğu ve her mekanın komşusuna 1/n değerini verdiği düşünülür. Belirli bir mekan için bu değerlerin toplamı o mekanın kontrol değerini verecektir. Kontrol değeri yüksek olan mekanlar komşuları ile daha güçlü, kontrol değeri düşük olan mekanlar komşularıyla daha zayıf bağlantılara sahiptir. Seçim (choice/ betweenness) değeri ise, bir mekanın diğer tüm mekanlar arasindaki en kısa yollarda çiğnenme sıklığıdır(Hillier ve diğ., 1987); bir mekan içinden akışı (flow) ölçer (Klarqvist, 1993).



Şekil Ek 12-11. Oluşturulan ağ yapısının entegrasyon analizi.

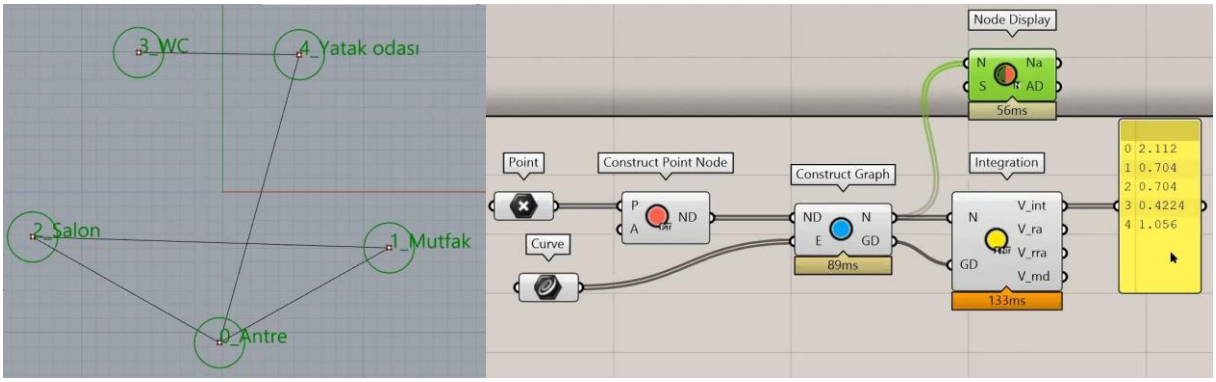
Bu hesaplamaları yapan Grasshopper bileşenlerinin kullanım şekli birbirine benzerdir. Örneğin *Integration* isimli bileşen, N ve GD girdileri *Construct Graph*’ın aynı isimli çıktılarına bağlanarak çalıştırılabilir (Şekil Ek 12-11).

Üretilen ağ yapısını görselleştirmek için SpaceChase sekmesi altındaki *Display* bileşen grubunda bulunan *Node Display* bileşeni kullanılabilir. Bu bileşenin girdisi olan N’ye *Construct Graph*’ın çıktısı olan N bağlandığında, Rhinoceros ekranında düğüm noktaları görselleştirilir (Şekil Ek 12-12). S girdisi, düğüm noktalarının ekrandaki görüntüsünün büyüklüğünü ayarlamak içindir.

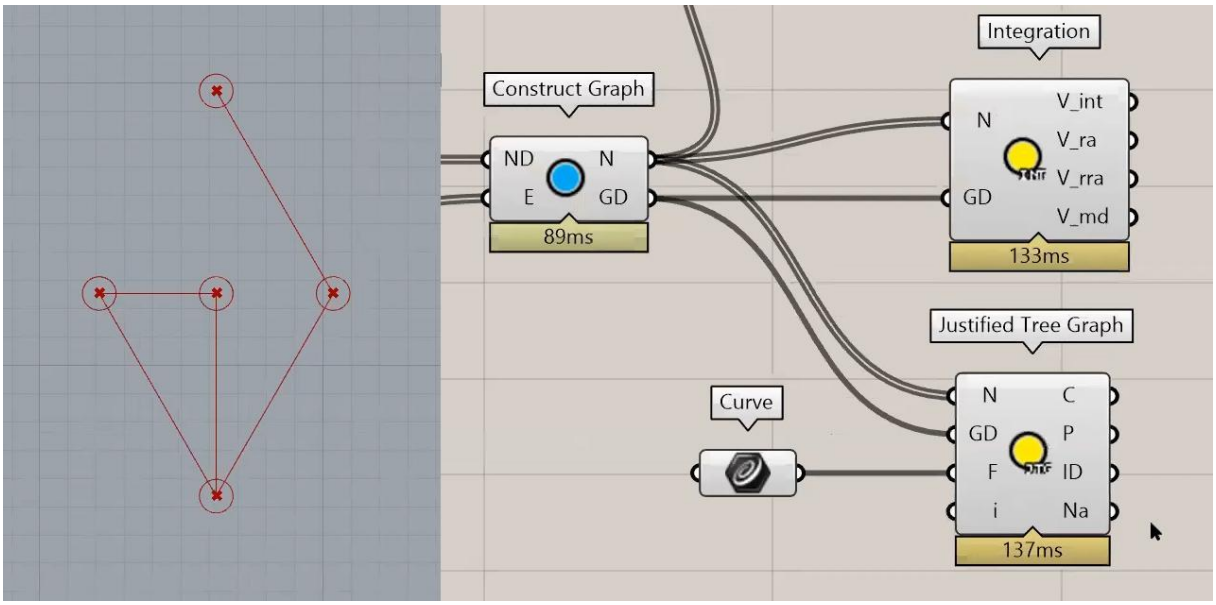


Şekil Ek 12-12. Node Display bileşeni ile temel ağ görselleştirmesi. Bu bileşen düğüm noktalarının isimlerinin ve sıra numaralarının Rhinoceros ekranında görüntülenmesini sağlar. Burada görüntülenen isimler Şekil Ek 12-3'te noktalara girilen isimlerdir.

Analiz sonuçlarını sayısal veri olarak gözlemlemek için Grasshopper'ın Panel bileşeni ilgili çıktılara bağlanmalıdır (Şekil Ek 12-13). Panel bileşenini kanvasa çift tıklayarak açılan arama kutucuğuna *Panel* yazmak suretiyle bulmak mümkündür. Şekil Ek 12-11'de gösterilen örnekte, sağ taraftaki *Panel* bileşeninde okunan entegrasyon değerleri, her bir düğüm noktasının (mekanın) sıra numaraları (index) ile eşleştirilmiştir. Örneğin, 0 indeks numarası olan "Antre" mekanının entegrasyon değeri *Panel*'deki 0 indeksli değer olan 2.112'dir. 2 indeks numaralı "Salon"un entegrasyon değeri *Panel'*deki 2 indeksli değer olan 0.704'tür.



Şekil Ek 12-13. Görselleştirilen düğümlerin entegrasyon değerlerinin okunması.



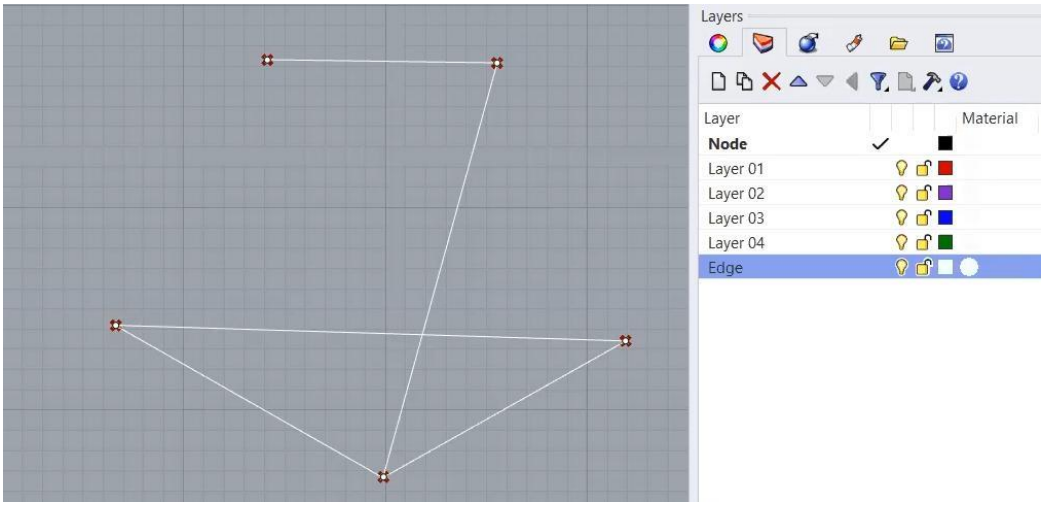
Şekil Ek 12-14. *Justified Tree Graph* oluşturulması. *Construct Graph* bileşenin çıktılarının ve çerçeve görevi gören bir dikdörtgenin (*Curve parameter* vasıtasıyla) *Justified Tree Graph* bileşenine bağlanması.

Justified Tree Graph bileşeni ile ağ yapısının görselleştirilmesi sağlanır. Bu bileşenin F girdisi, grafiğin oluşturulacağı dikdörtgen çerçeveyi içerir. F girdisi üzerinde sağ tıklanarak *Set One Rectangle* seçeneği ile Rhinoceros'ta önceden çizilmiş bir dikdörtgen çerçeve olarak kullanılabilir.

Bu bileşenin *i* girdisi ise hangi düğüm noktasıdan grafiğin hesaplanacağını belirleyen girdidir (Şekil Ek 12-14). Bu bileşenin çıktıları kullanılarak farklı görselleştirme alternatifleri oluşturulabilmektedir. Örneğin, Grasshopper'in *Text Tag 3D* isimli bileşeni yardımıyla farklı görsel sonuçlar elde edilebilmektedir.

4. Dinamik Ağ Yapıları

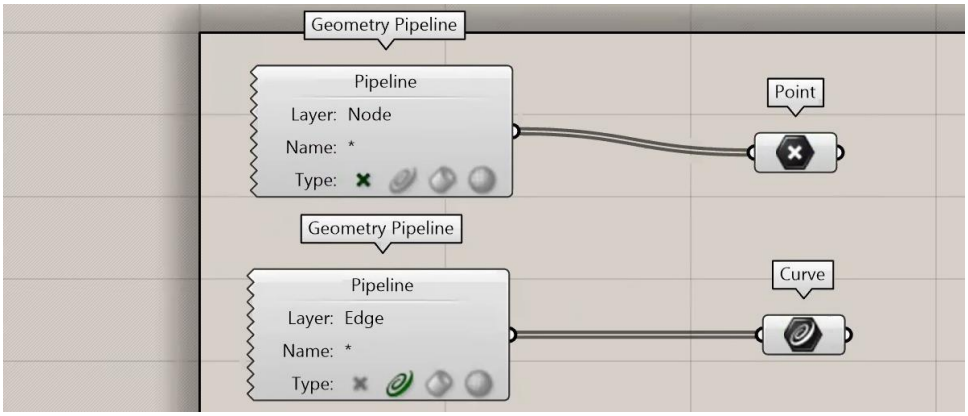
Bu bölümde bir önceki bölümde Rhinoceros'ta üretilen nokta ve bağlantılar kullanılacaktır. Dinamik ağ yapıları, önceki bölümde açıklanan temel kullanım biçimine ek olarak düğüm noktalarına ve bağlantılara bazı yeni özellikler katmaktadır. Bu farkları ayırt edebilmek için dinamik ağ yapılarında Rhinoceros'un katmanlarının (*Layer*) aktif olarak kullanılması önerilmektedir. Örneğin, *Nodes* isimli bir katman ve *Edges* isimli bir başka katman oluşturularak ilgili nesneler bu katmanların içerisine alınabilir (Şekil Ek 12-15). Rhinoceros'taki bir nesnenin katmanını değiştirmek için, o nesne seçili iken katmanın isminin üzerinde sağ tıklanarak *Change Object Layer* seçeneği kullanılabilir.



Şekil Ek 12-15. Dinamik ağ yapısına hazırlık amacıyla katmanların oluşturulması.

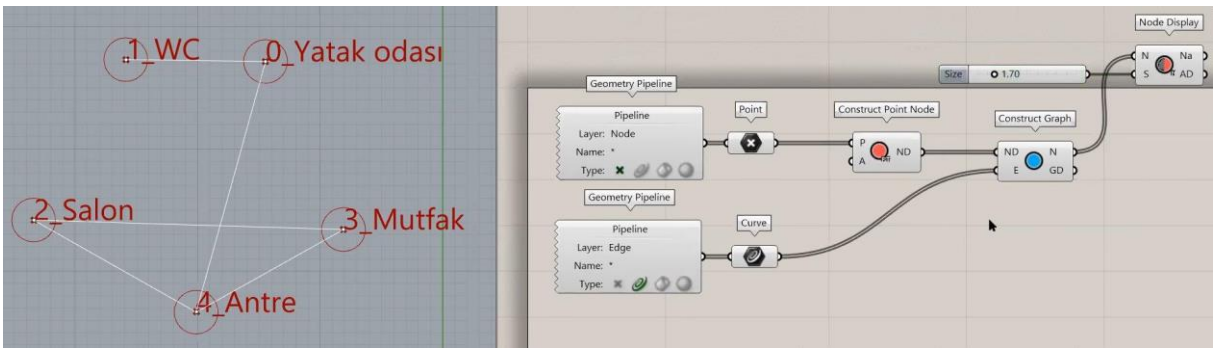
Dinamik ağ ortamında etkileşimi artırmak için Grasshopper'ın *Geometry Pipeline* prosedürünün kullanımı önerilmektedir. *Geometry Pipeline* isimli bileşen aranarak kanvasa yerleştirildikten sonra, şekilde görüldüğü gibi, belirli katmanlardaki ve belirli türdeki (nokta, çizgi vb.) nesneleri gerçek zamanlı olarak okuması ve Grasshopper içerisine aktarması sağlanabilir.

Böylelikle, dinamik ağ simülasyonu sırasında sistemi yeniden başlatmadan müdahale etmek mümkün hale gelir (Şekil Ek 12-16).



Şekil Ek 12-16. *Geometry Pipeline* bileşeni ile *Node* ve *Edge* isimli katmanlardaki nesnelerin okunması ve birer *Point ve Curve Parameter* yardımıyla Grasshopper içerisine aktarılması.

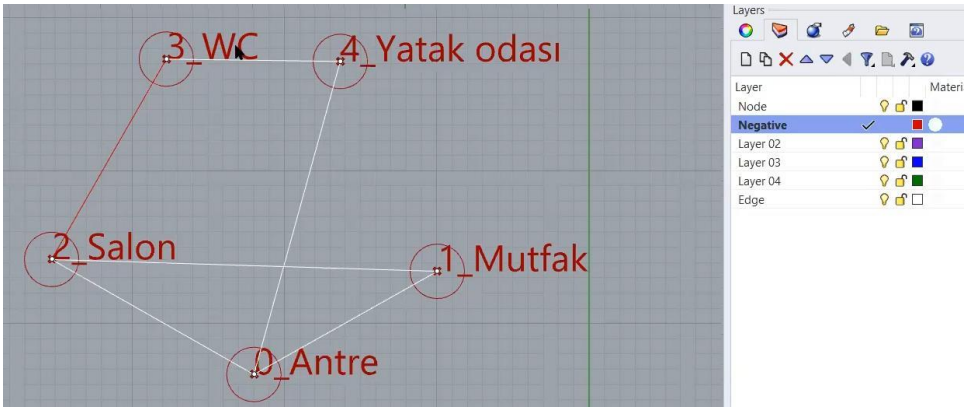
Bu düzenlemenin dışında dinamik ağ yapısının hazırlığı bir önceki bölümde anlatılan durağan ağ yapısı ile aynıdır. Oluşturulan *Node Data* (ND) ve *Graph Data* (GD) verileri dinamik ağ yapısında da kullanılacaktır. Şekil Ek 12-17'de hazırlık aşamasının sonucu gösterilmiştir.



Şekil Ek 12-17. Dinamik ağ yapısına hazırlık amacıyla oluşturulan temel kurulum. Rhinoceros ve Grasshopper ekranları.

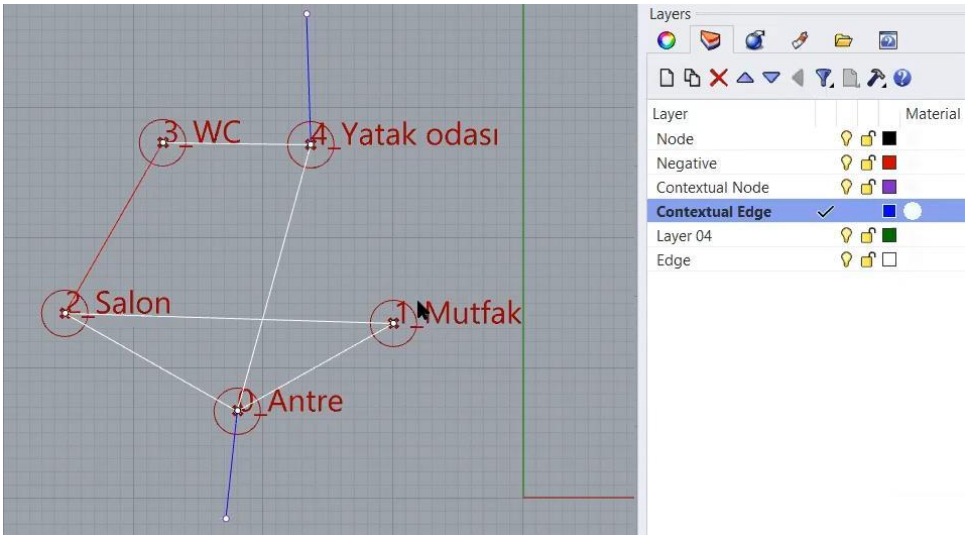
Dinamik ağ yapısında düğüm noktalarının bir simülasyon içerisinde devinime girmesi beklenmektedir. SpaceChase içerisinde bu devinimi kontrol eden ve düğümler arasındaki itme ve çekme etkilerini belirleyen ek bilgilerin girilmesi mümkündür. Dinamik ağ yapılarında bağlantılar, aynı zamanda ağın hareketini de belirleyen unsurlardır. Dolayısıyla, SpaceChase içerisinde negatif ve pozitif bağlantılar tanımlanabilmektedir. Negatif değerdeki bağlantılar, iki ucundaki düğüm noktalarını birbirinden uzaklaştırma eğilimindedir. Şekil Ek 12-18'de Rhinoceros'ta ilgili katmanın açılmasının ardından iki düğüm noktası arasına bir negatif bağlantının eklenmesi görülmektedir. Bu tip bağlantıların farklı biçimde tanımlanmasının bir nedeni, bunların Mekan Dizimi analizlerine birer mekansal bağlantı olarak girmelerinin önüne

geçmektir. Dolayısıyla bu bağlantılar *Construct Graph* bileşeninin E girdisine katılmayacaklardır.



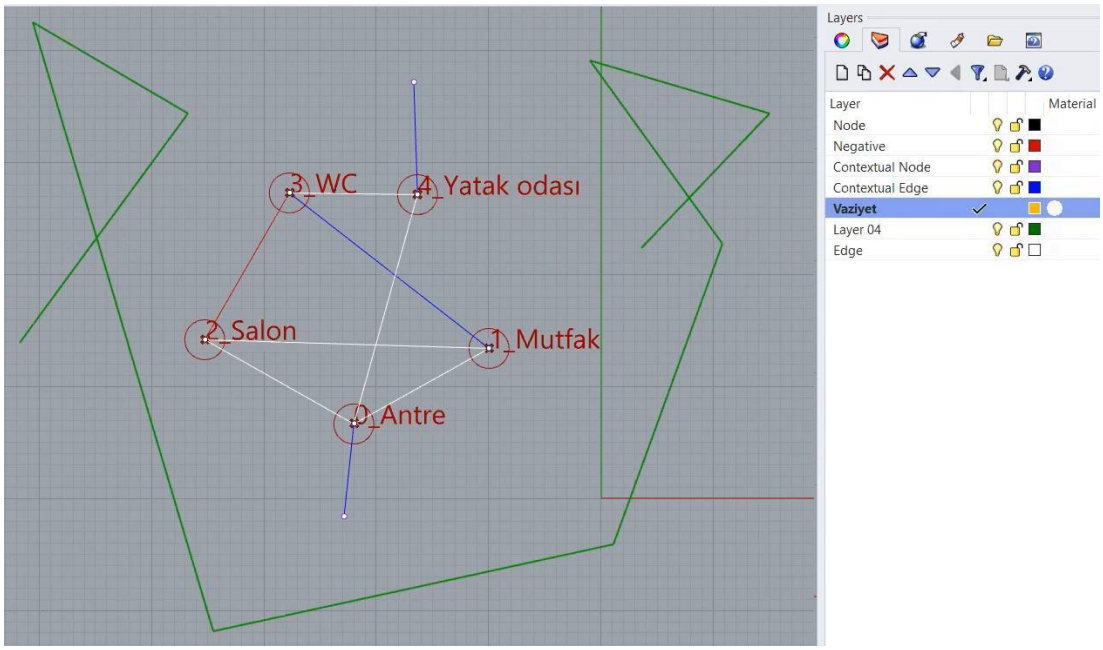
Şekil Ek 12-18. Rhinoceros'ta negatif bir bağlantının tanımlanması. WC ve Salon isimli düğüm noktalarının arasına kırmızı renkli *Negative* isimli katmanın içerisinde bir çizgi çizilmiştir.

Dinamik ağın hareketini kontrol edebilmek için negatif bağlantılara ek olarak başka araçlar da vardır. Hareket sırasında sabit konumda kalması istenen ve sentaktik hesaplamalara katılmayan, sadece diğer düğüm noktalarının konumunu ve hareket biçimini etkilemesi istenen özel düğüm noktaları tanımlanmıştır. Bu noktalar bağlamsal düğümler *(Contextual Node)* olarak isimlendirilmiştir. Bağlamsal düğümler, prototip aşamasından itibaren yazılım eklentisinde bulunan özgün bir konsepttir. Bağlamsal düğüm noktaları da normal düğüm noktaları gibi Rhinoceros'ta birer nokta olarak çizilir. *Contextual Node* isimli bir katmanda saklanmaları önerilmektedir. Bağlamsal düğüm noktalarının seçilen düğüm noktalarını kendilerine çekmeleri beklenir. Bu çekme etkisi, yine sentaktik hesaplamalara girmeyen, pozitif bağlantıların tanımlanmasını gerektirmiştir. Negatif bağlantılarda olduğu gibi pozitif ilişki (çekme etkisi) gösteren bu bağlantılar da Rhinoceros ekranında tanımlanabilirler. Bu bağlantılar bağlamsal bağlantı (*Contextual Edge*) olarak isimlendirilmiştir. Şekil Ek 12-19'da iki bağlamsal düğüm noktası ve bu düğümlerden iki normal düğüm noktasına uzanan iki bağlamsal bağlantının tanımlanması gösterilmiştir. Ağın üst tarafında tanımlanan bağlamsal düğüm noktası, Rhinoceros içerisinde "Kuzey" olarak isimlendirilmiş ve *Contextual Node* isimli bir katmanının içerisinde tanımlanmıştır. Ağın alt tarafında tanımlanan bağlamsal düğüm noktası da aynı katman içerisindedir ve yine Rhinoceros'ta "Yol" olarak isimlendirilmiştir (Bkz. Şekil Ek 12-3). "Kuzey" bağlamsal düğüm noktası ile "Yatak Odası" düğüm noktası arasında *Contextual Edge* isimli katmanda çizgi çizilerek bir bağlamsal bağlantı tanımlanmıştır. Aynı şekilde, "Yol" bağlamsal düğüm noktası ile "Antre" düğüm noktası arasında da bir bağlamsal bağlantı tanımlanmıştır.



Şekil Ek 12-19. Bağlamsal düğüm noktalarının ve bağlamsal bağlantıların (pozitif) tanımlanması.

Negatif bağlantıların veya bağlamsal bağlantıların normal düğümler arasında tanımlanması da mümkündür. Pozitif (bağlamsal) ve negatif bağlantılar ve bağlamsal düğüm noktaları sentaktik analizlere girmezler; dolayısıyla sadece simülasyonun hareket biçimini belirlemek için kullanılırlar. Bu nesnelerin SpaceChase içerisinde kullanılabilmeleri için Grasshopper içerisine aktarılmaları gerekir.



Şekil Ek 12-20. Vaziyet isimli bir katman içerisine ağ ile ilgili olmayan diğer çizimlerin yerleştirilmesi. Bu nesneler Grasshopper kanvası içerisine alınırken *Curve Parameter*, veya *Geometry Parameter* bileşenleri kullanılabilir.

Bu nesneler Şekil Ek 12-20'de görüldüğü gibi farklı katmanlarda oluşturulduysa bu işlem için yine *Geometry Pipeline* bileşeninin kullanılması önerilmektedir. SpaceChase ile doğrudan etkileşime girmeyecek diğer çizimler, her türden (nokta, çizgi, vb.) geometrik nesneyi içerebildiği için *Geometry Parameter* isimli genel amaçlı konteynır bileşenine alınmalıdır. Şekil Ek 12-21'de Vaziyet isimli bir katman açılarak bunun içerisine alınan çeşitli geometrik nesneler görülmektedir.



Şekil Ek 12-21. Rhinoceros ekranının hareketli ağ yapısı için ayarlanması, Rhinoceros çizimlerinin y ekseninin bir tarafında toplanması. Bu örnekte çizimler eksenin sol tarafında toplanmıştır. Bu durumda eksenin sağ tarafı SpaceChase tarafından hareketli ağın görselleştirilmesinde kullanılacaktır.

SpaceChase, yukarıda açıklanan ayarlamaların ve geometrik nesnelerin (Rhinoceros noktaları ve çizgileri) hareket esnasında daha rahat kontrol edilebilmesi için Rhinoceros ekranını y ekseni (yeşil eksen) boyunca ayırarak iki ekran halinde kullanmaktadır. Dolayısıyla yapılan çizimlerin bu eksenin bir tarafına alınması önerilmektedir. Şekil Ek 12-21'de bu ayrım için yapılan ön hazırlık gösterilmiştir.

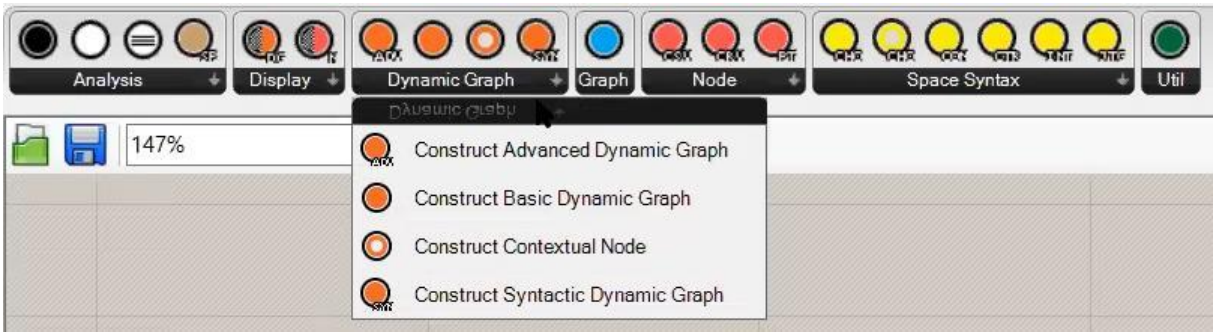
Yukarıdaki ön hazırlıklar tamamlandıktan sonra ağ yapısının dinamik simülasyonuna geçilebilir. SpaceChase'in *Dynamic Graph* başlıklı bileşen grubu içerisinde bulunan bileşenler farklı amaçlar için çözümler sunmaktadır (Şekil Ek 12-22). Üç çeşit dinamik ağ modeli bulunmaktadır. Bunlar;

***Basic*:** Sadece düğüm noktaları, bağlantılar, negatif bağlantılar, pozitif bağlantılar (*Contextual Edge*) ve bağlamsal düğümler (*Contextual Node*) içeren simülasyonları gerçekleştirmek için kullanılır. Bu model, hızlı dinamik ağ görselleştirmesine yönelik kullanılmaktadır. Grasshopper ortamında yapılamayan döngüsel hesaplamaları kendi içerisinde yapan bu bileşenlerden ilki

çizgi ve nokta verilerini pozitif/negatif ilişkiler, uzunluk ve mesafeler bakımından hesaplayarak bir dinamik graf modeli üretir.

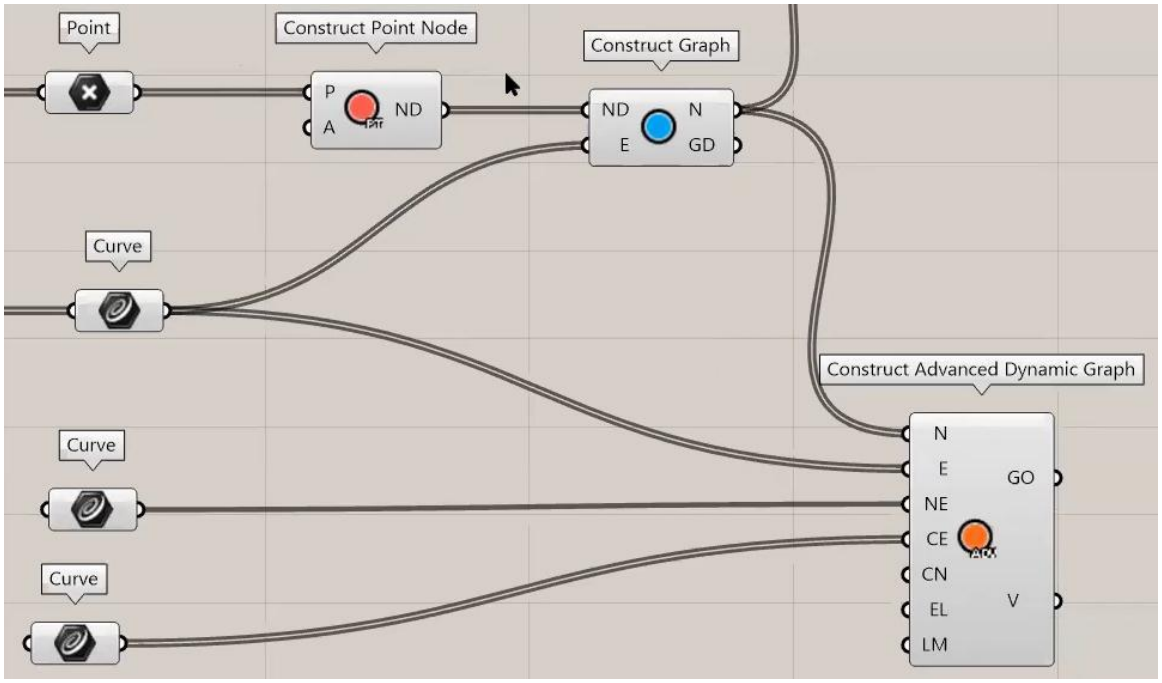
***Advanced*:** *Basic* ağ yapısına ek olarak, bağlantılar arasındaki mesafeleri ayrı ayrı belirlemeye olanak tanır. Ayrıca, *Link Manager* adı verilen bir arayüze sahiptir ve bu sayede düğümler arasındaki ilişkileri daha detaylı biçimde belirlemeye olanak tanır. Basic bileşenden farklı olarak vektörel çıktıya da sahiptir. Bu ağ modeli hesaplamalara bireysel veya toplu olarak müdahale edilebilme olanağı sağlar. Benzer hesaplamalar yapan ancak müdahale olanağını bir arayüz vasıtasıyla kullanıcı dostu bir şekilde sunabilmek için sıfırdan kodlanan ikinci graf modeli, kullanıcı tarafından hazırlanmasının daha fazla vakit gerektirmesi nedeniyle birinci modelden ayrı tutulmuştur.

***Syntactic*:** *Advanced* ağ yapısına ek olarak, ağırlık (*Weight*) ve güç (*Strength*) parametrelerine sahiptir. Düğümlerin ağırlığı ve ilişki gücü bu yolla kontrol edilebilir. Örneğin, bir önceki bölümde kullanım şekli anlatılan Mekan Dizimi metrikleri bu yolla düğümlerin ağırlık ve ilişki güçlerine bağlanarak bu metriklerin simülasyon üzerinde etkili olmaları sağlanabilir. Bu ağ modeli ikinci modelde kullanıcının insiyatifinde olan hesaplamaları, bileşene girdi olarak seçilen metrikler yardımıyla yapar ve "sentaktik" bir dinamik ağ görselleştirmesi olanağı sunar.



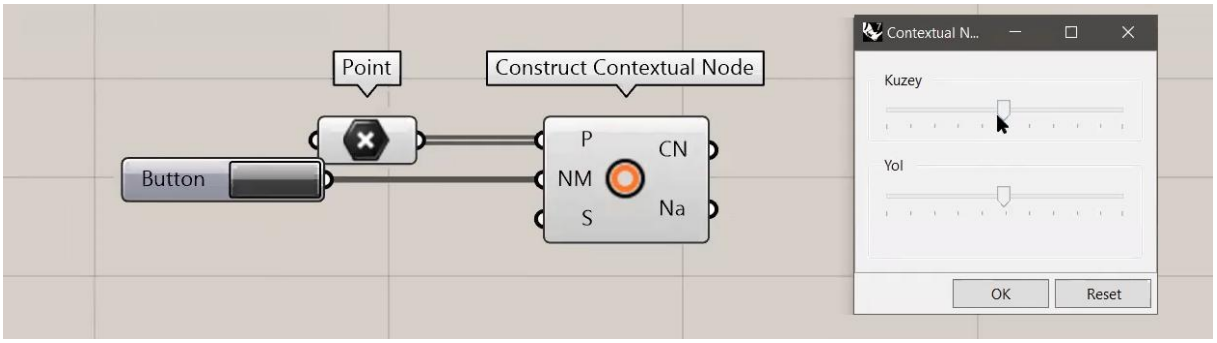
Şekil Ek 12-22. SpaceChase dinamik ağ modellerini oluşturan bileşen seçenekleri.

Bu kılavuzun devamında *Advanced* ağ modeli kullanılacaktır. Bunun için *Construct Advanced Dynamic Graph* bileşeni kanvasa yerleştirilip bağlantıları yapılmalıdır. Bu bileşenin girdilerine sırasıyla, Construct Graph bileşeninden çıkan Node (N), bağlantı çizgilerini ifade eden Edge (E), Grasshopper içerisine aktarılmış negatif bağlantılar (NE), ve bağlamsal bağlantılar (CE) bağlanmalıdır. Şekil Ek 12-23'te bu bağlantılar gösterilmiştir.



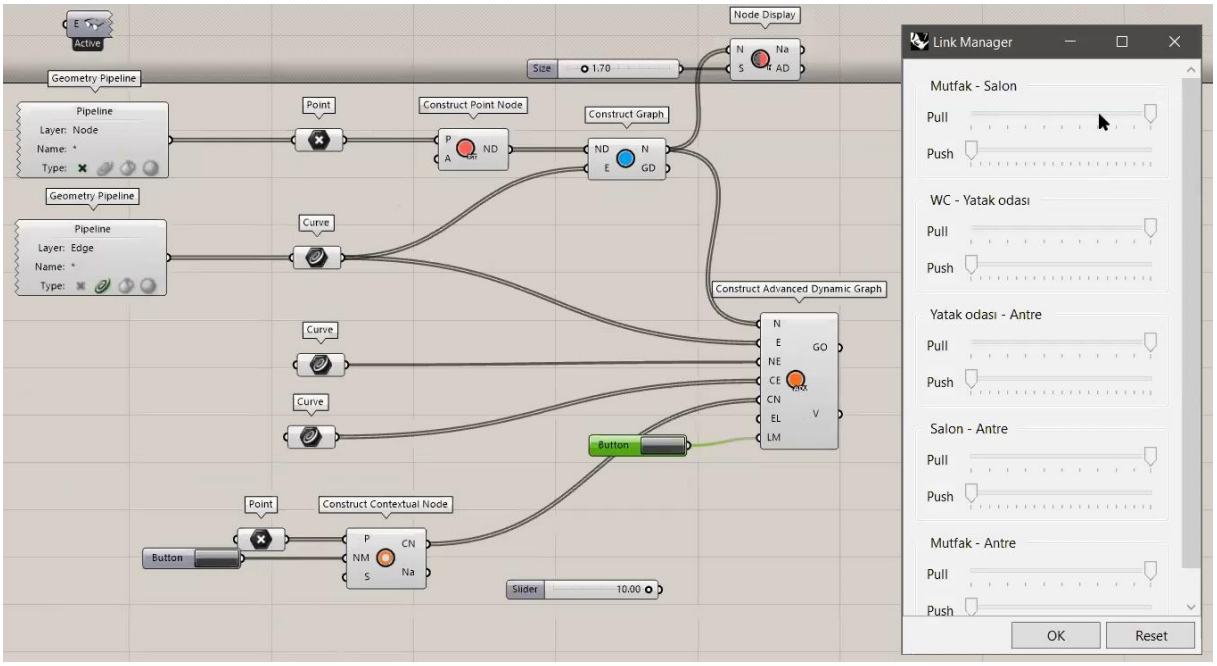
Şekil Ek 12-23. C*onstruct Advanced Dynamic Graph* bileşeninin temel bağlantıları. Bu örnekte *Curve Parameter* bileşenleri kullanılarak Rhinoceros'tan aktarılan negatif bağlantıları (NE), ve bağlamsal (pozitif) bağlantıları (CE) *Geometry Pipeline* kullanarak aktarmak da mümkündür.

Construct Advanced Dynamic Graph bileşenine bağlamsal düğüm noktalarının (*Contextual Node*) aktarımı ise bir ara bileşen yardımıyla yapılmaktadır. Bu düğüm noktaları doğrudan Rhinoceros'tan alınabilecek nokta nesneleri olmanın ötesinde birer düğüm (*Node*) olma niteliği de taşırlar. Bu nesneleri SpaceChase'in işleyebileceği bağlamsal düğümlere dönüştürmek için *Dynamic Graph* bileşen grubunun altındaki *Construct Contextual Node* bileşeni kullanılır. Bağlamsal düğüm noktası, diğer SpaceChase düğüm bileşenlerinde olduğu gibi bir Point Parameter vasıtasıyla bu bileşene bağlanabilmektedir. Normal düğüm noktalarından farklı olarak bu bileşenin NM isimli bir girdisi bulunmaktadır. Bu girdiye Grasshopper'ın Button isimli bileşeni bağlanmalıdır. Bu düğme görevi gören bir bileşendir. Bu bileşen NM girdisine bağlandıktan sonra düğmeye basıldığında Contextual Node Manager isimli bir pencere açılacaktır. Şekil Ek 12-24'te görülen örnek, Rhinoceros'ta tanımlanan ve "Kuzey" ve "Yol" biçiminde isimlendirilen iki noktanın bir *Point Parameter* ile Grasshopper içerisine alındıktan sonra *Construct Contextual Node*'un P girdisine bağlanmasını ve ardından NM girdisine bağlanan bir Button bileşenine basıldığında açılan pencereyi göstermektedir.



Şekil Ek 12-24. *Construct Contextual Node* bileşeni ve *Contextual Node Manager* penceresi.

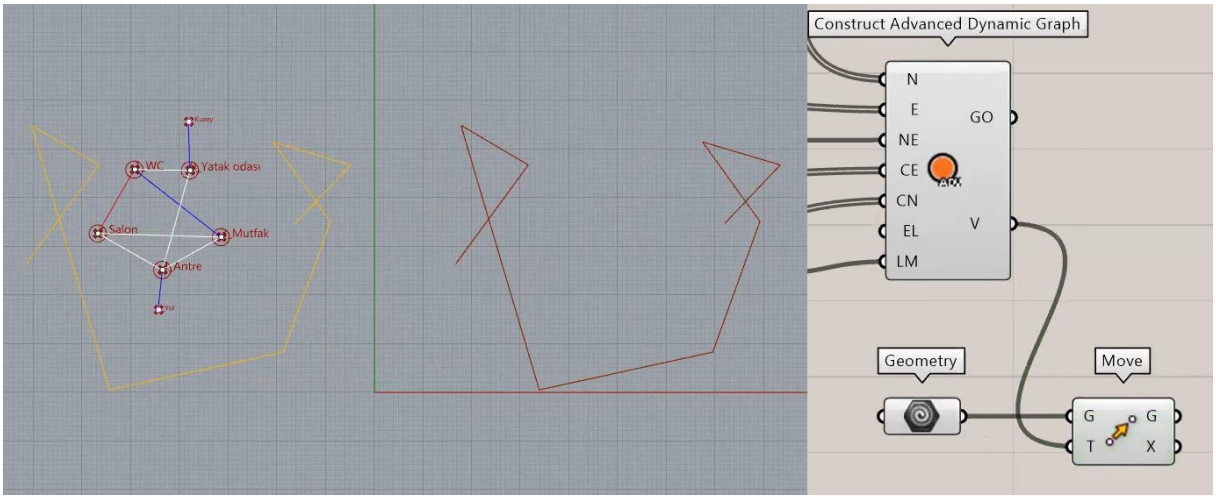
Contextual Node Manager'de bulunan kaydırma çubuğu ile bağlamsal düğüm noktalarının etki mesafelerini düzenlemek mümkün olacaktır. *Construct Contextual Node* bileşeninin CN çıktısı, *Construct Advanced Dynamic Graph* bileşeninin CN girdisine bağlanmalıdır. Bu bileşenin son iki girdisinden biri olan *Edge Length* (EL) ağ yapısındaki bütün bağlantıların uzunluklarını belirli bir sayı değerine eşitlemek için kullanılan opsiyonel bir girdidir. *Link Manager* (LM) girisine de bir *Button* bileşeni bağlanmalıdır. Bu girdi, düğmeye basılıp aktifleştirildiğinde *Contextual Node Manager*'de olduğu gibi *Link Manager* isimli bir ayarlama penceresi açacaktır. Bu pencere kullanılarak ağ yapısındaki bütün bağlantıların itme ve çekme etkileri hassas biçimde kontrol edilebilmektedir.



Şekil Ek 12-25. *Construct Advanced Dynamic Graph* bileşeninin bağlantıları ve *Link Manager* penceresi.

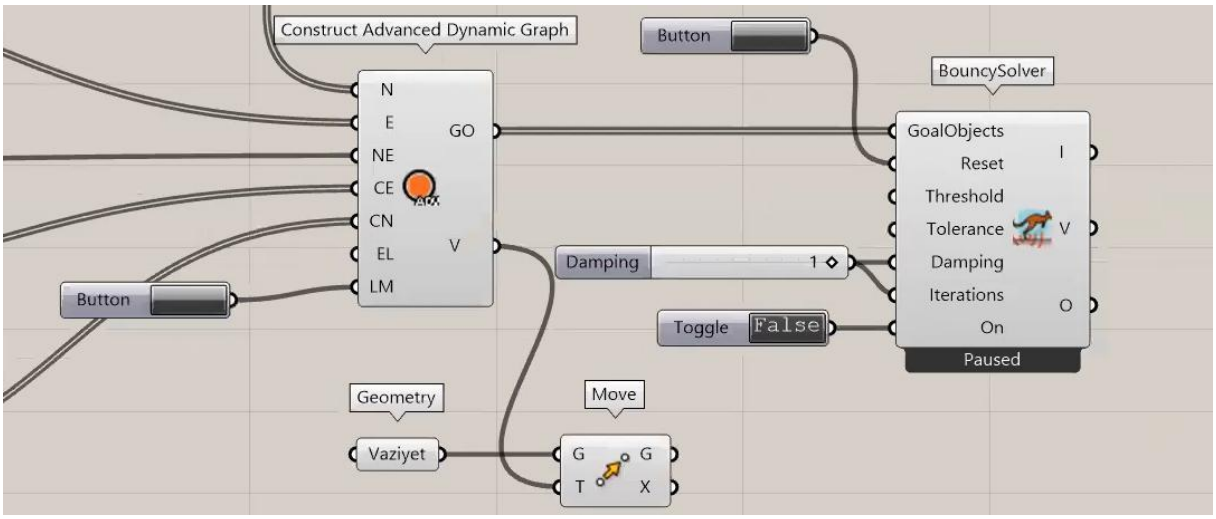
Şekil Ek 12-25'te *Link Manager* penceresi ve *Construct Advanced Dynamic Graph* bileşeninin kalan bağlantıları gösterilmiştir. *Link Manager* penceresinde bulunan *Reset* düğmesi ile bu

pencere üzerinde yapılan ayarlamalar iptal edilebilir. Bu bağlantılar ile *Construct Advanced Dynamic Graph* bileşeni hareketli ağ için gereken bütün girdileri toplamış olacaktır. Bu bileşenin birinci çıktısı olan GO, Kangaroo hedefleridir. Diğer çıktısı olan V ise, daha önce açıklanan ikili ekran kullanımına yardımcı olan bir vektördür. Bu çıktı kullanılarak hareketli ağ yapısında görülmesi istenen (vaziyet planı gibi) çizimlerin aktarımı sağlanabilmektedir. Bunun için Grasshopper'ın *Move* bileşeni kullanılabilir (Şekil Ek 12-26).



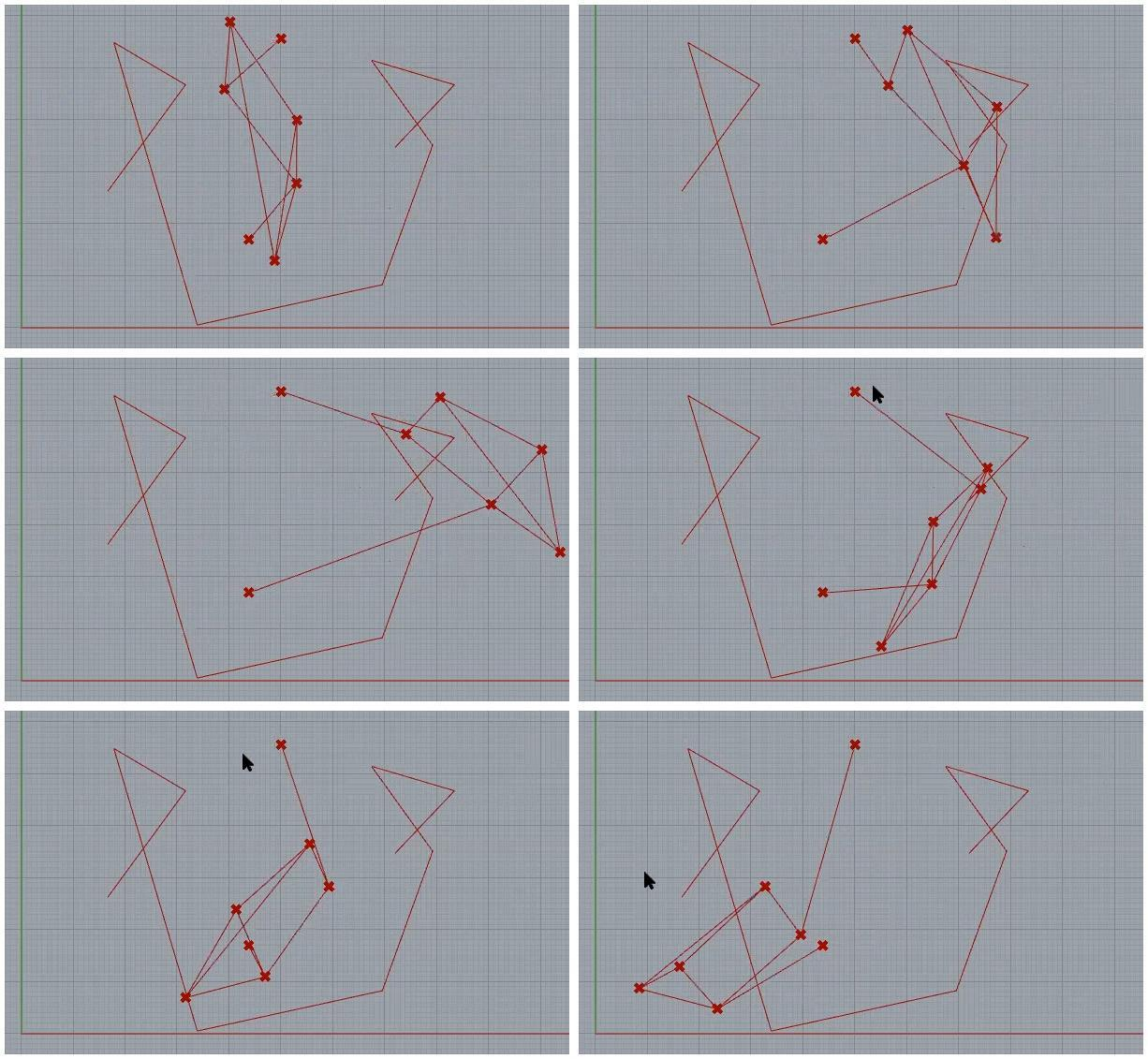
Şekil Ek 12-26. Hareketli ağ ile doğrudan etkileşime girmesi istenmeyen çizimlerin hareketli ağın görüntüleneceği ekran bölgesine taşınması.

Hareketli ağın oluşturulması için son adım, Grasshopper'ın Kangaroo bileşenleri içerisindeki *Bouncy Solver* bileşeninin kanvasa yerleştirilmesidir. Bu bileşenin *Goal Objects* girdisine, *Construct Advanced Dynamic Graph* bileşenin GO çıktısı bağlanmalıdır. *Reset* girdisine bir *Button* bileşeni, *On* girdisine de Grasshopper'ın *Toggle* bileşeni bağlanmalıdır.



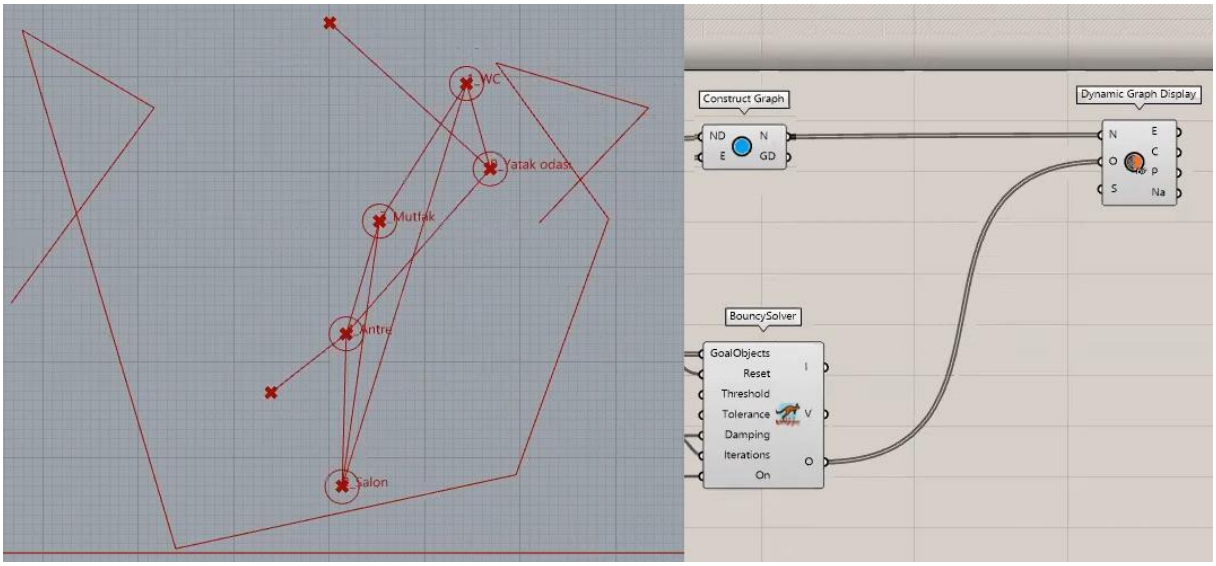
Şekil Ek 12-27. Kangaroo *Bouncy Solver* bileşenine girdilerin aktarılması ve hareketli ağ için ayarlanması.

Damping ve *Iterations* girdilerine "1" değeri bağlanmalıdır. Bu girdi bir *Panel* veya *Number Slider* bileşeni kullanılarak yapılabilir. Bu bağlantılar Şekil Ek 12-27'de gösterilmiştir. *Bouncy Solver* bileşeninin *On* girdisine bağlı *Toggle* üzerine çift tıklanarak *True* pozisyonuna getirildiğinde Rhinoceros ekranında hareketli ağ yapısı oluşmalıdır. Şekil Ek 12-28, bu hareketin bir örneğini göstermektedir. Yapılan örnekte çizimler y ekseninin sol tarafına alındığı için SpaceChase y ekseninin sağ tarafına doğru yaylanarak hareket eden bir ağ oluşturmuştur. İki bağlamsal düğümün çıpa etkisi yaparak ağın ekran dışına çıkmasını önlediği gözlemlenebilir. Ayrıca bağlantıların da hareket sırasında sabit kaldığı gözlemlenebilir. Bu hareket esnasında görüntülenen bağlantılar sentaktik olan ve olmayan bütün bağlantıları içermektedir. Bu temel kurulum üzerinde çalışmaya devam edilerek tasarım senaryosuna ve kullanım amacına uygun Grasshopper algoritmaları geliştirmek mümkündür.



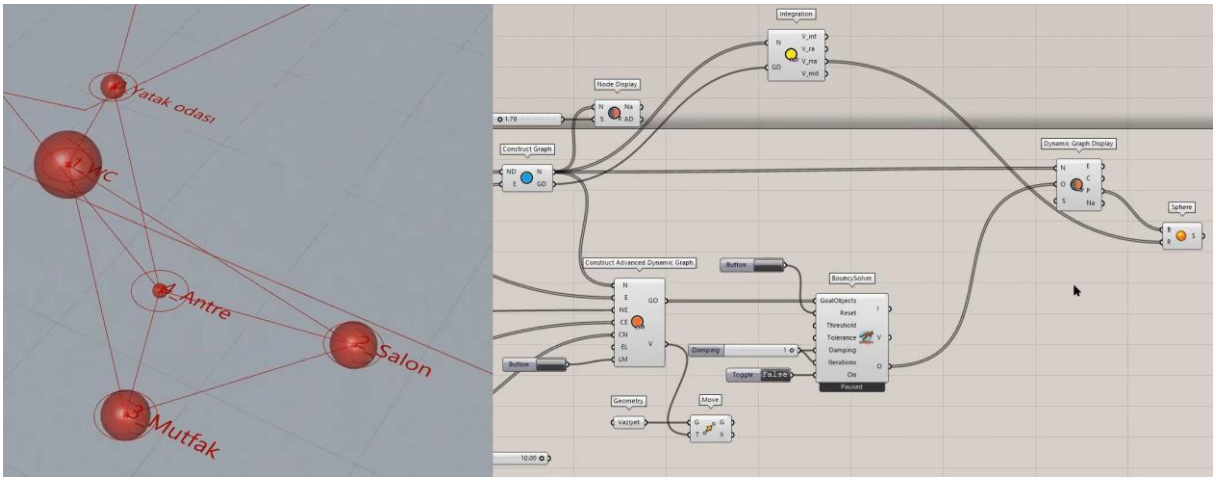
Şekil Ek 12-28. Örnek ağ yapısının kazandığı ilk hareket.

Şekil Ek 12-28'de görüntülenen ilk harekette bağlamsal düğümler haricinde diğer düğümlerin hangileri olduğunu anlamak mümkün değildir. Hareketi daha anlaşılır hale getirmek için SpaceChase sekmesinde *Display* bileşen grubu altındaki *Dynamic Graph Display* isimli bileşen kullanılmalıdır. Bu bileşenin N girdisi, Construct Graph bileşeninin N çıktısına, O girdisi ise Bouncy Solver bileşeninin O çıktısına bağlanmalıdır. Böylelikle Kangaroo'nun işlediği hareket bilgisi, SpaceChase'in Node bilgisi ile bütünleştirilmektedir. Bu bileşen, Şekil Ek 12-29'da görüldüğü gibi düğüm noktalarının isimlerini görüntülemektedir.



Şekil Ek 12-29. *Dynamic Graph Display* bileşeninin kullanım şekli.

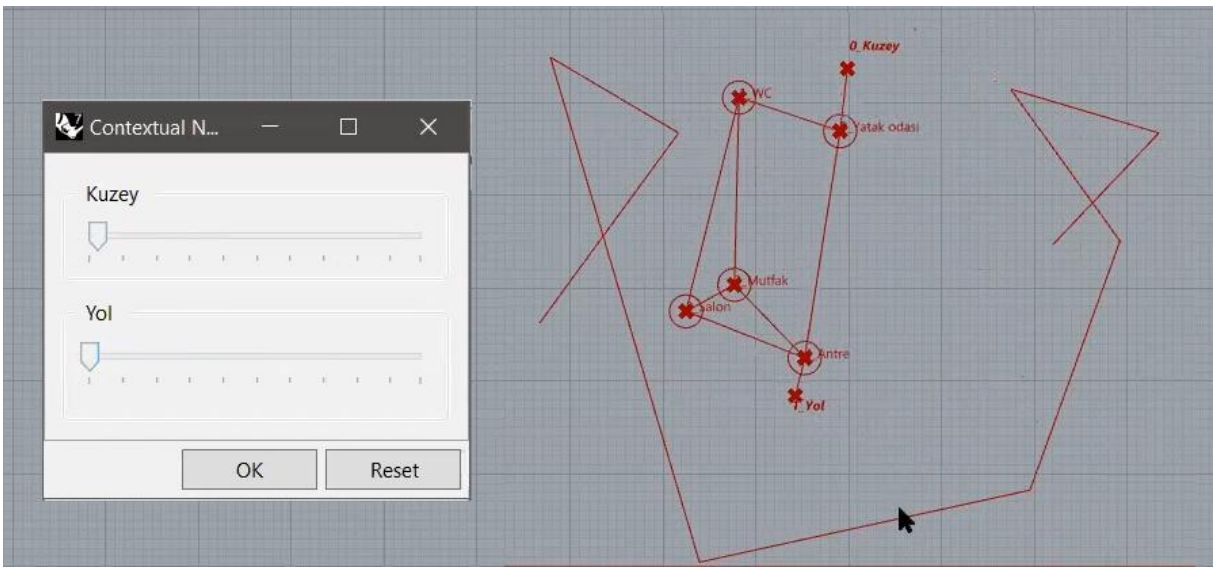
Bu bileşenin özgün farkı, farklı görselleştirme biçimlerine olanak sağlayacak biçimde bütün çıktılarını Grasshopper nesneleri biçiminde sunmasıdır. Bu çıktılar kullanılarak örneğin Şekil Ek 12-30'da görüldüğü gibi farklı görselleştirmeler tasarlanabilmektedir.



Şekil Ek 12-30. *Dynamic Graph Display* bileşeninin farklı Grasshopper bileşenleri ile beraber kullanımı.

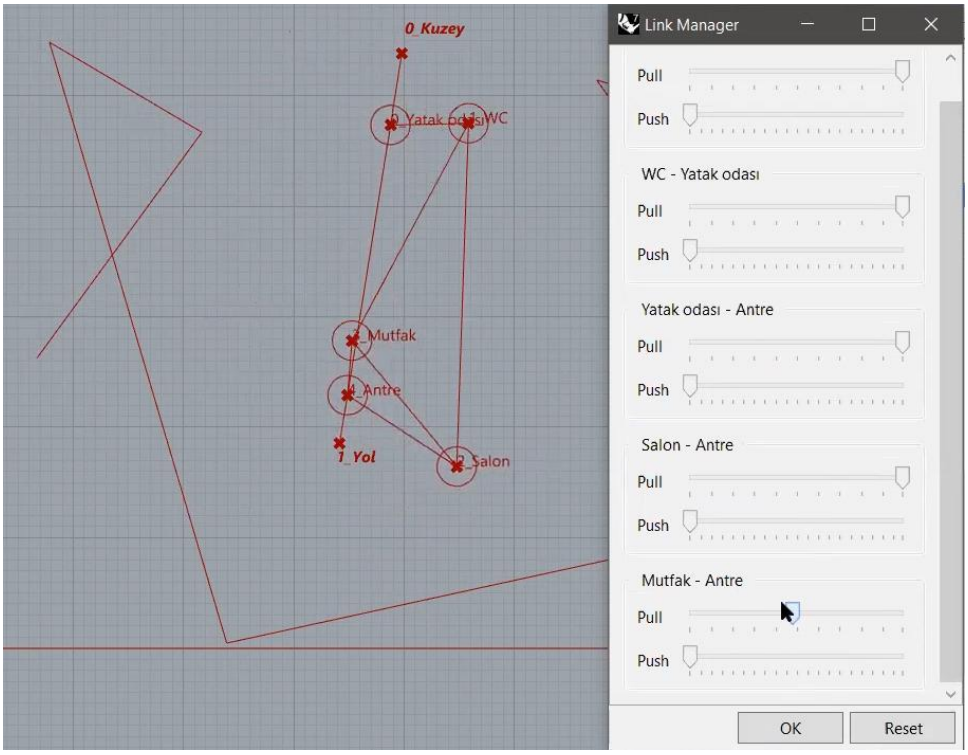
Şekil Ek 12-30'da, *Dynamic Graph Display* bileşeni kullanılarak, düğüm noktalarına birer küre çizilmiştir. Bunun için Grasshopper'ın *Sphere* bileşeni kullanılmıştır. Kürelerin yarıçapları ise her düğüm noktasının entegrasyon değerlerinden alınmaktadır.

Daha önce kurulumu yapılan *Contextual Node Manager* penceresi kullanılarak bağlamsal düğümlerin ağ üzerindeki etkileri kontrol edilebilmektedir. Bunun için *Construct Contextual Node* bileşeninin NM girdisine bağlanan düğmeye basılmalıdır. Şekil Ek 12-31'deki örnek, "Kuzey" isimli bağlamsal düğümün "Yatak Odası" isimli düğümü kendine daha fazla çektiği, aynı şekilde, "Yol" isimli bağlamsal düğümün de "Antre" isimli düğümü kendine daha fazla çektiği ayarlamayı göstermektedir. Ağın hareketi gerçek zamanlı olarak bu girdiye tepki verdiği ve ilgili düğümlerin bağlamsal düğümlere yakın olduğu olasılıkları test ettiği gözlemlenecektir.

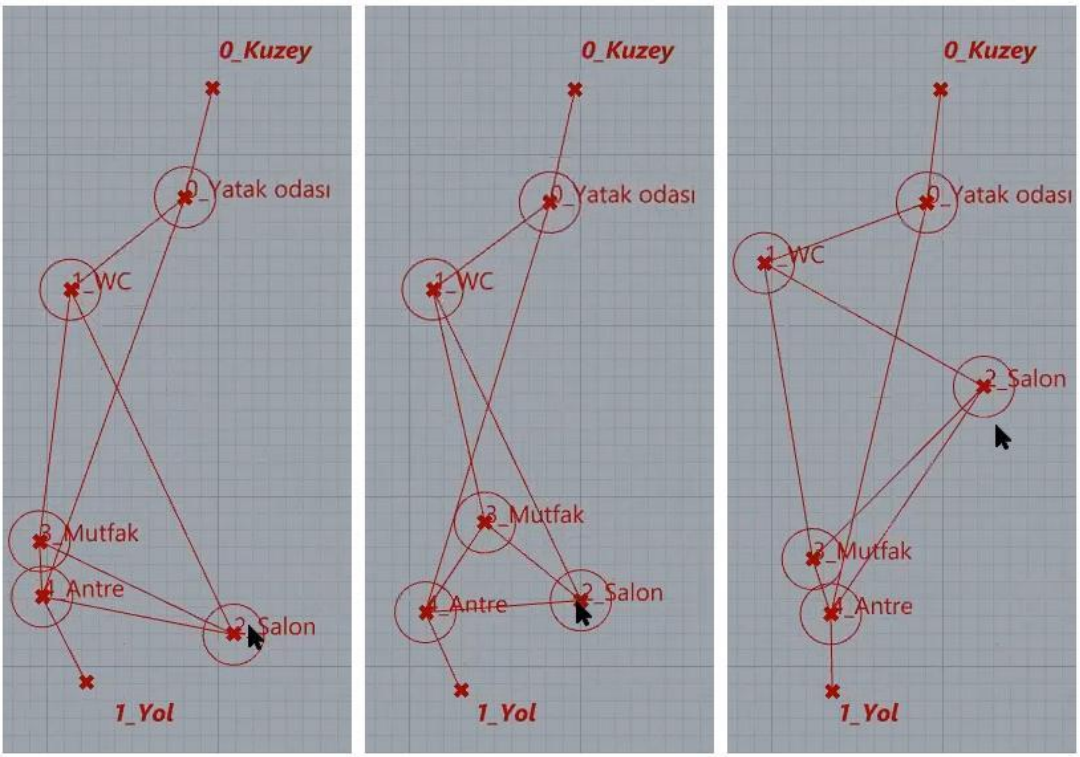


Şekil Ek 12-31. *Contextual Node Manager*'in hareketli ağ üzerinde gerçek zamanlı olarak kullanılması.

Benzer şekilde *Link Manager* isimli pencere de ağ hareket halindeyken kullanılarak farklı olasılıkların araştırılmasına yardımcı olacaktır. Bu pencere *Construct Advanced Dynamic Graph* bileşenin LM girdisine bağlanan düğmeye basılarak açılabilir. Şekil Ek 12-32'de yapılan örnek, "Mutfak" isimli düğüm noktası ile "Antre" isimli düğüm noktası arasında var olan sentaktik bağlantının çekme etkisinin arttırıldığı ve bu iki düğüm noktasının birbirine yaklaştırıldığı durumu göstermektedir. Bu penceredeki *Pull* isimli kaydırma çubuğu çekme etkisini, *Push* isimli kaydırma çubuğu da itme etkisini kontrol etmektedir.



Şekil Ek 12-32. Link Manager arayüzü kullanılarak düğüm noktaları arasındaki ilişkilerin daha hassas biçimde kontrol edilmesi.

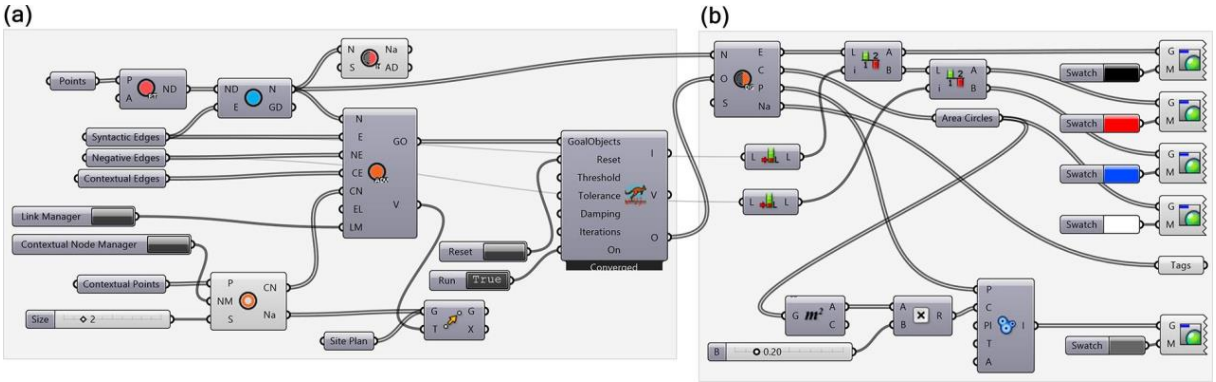


Şekil Ek 12-33. Hareketli ağ yapısına elle müdahale etmek mümkündür. Bunun için fare ile düğüm noktaları tutulup sürüklenmelidir.

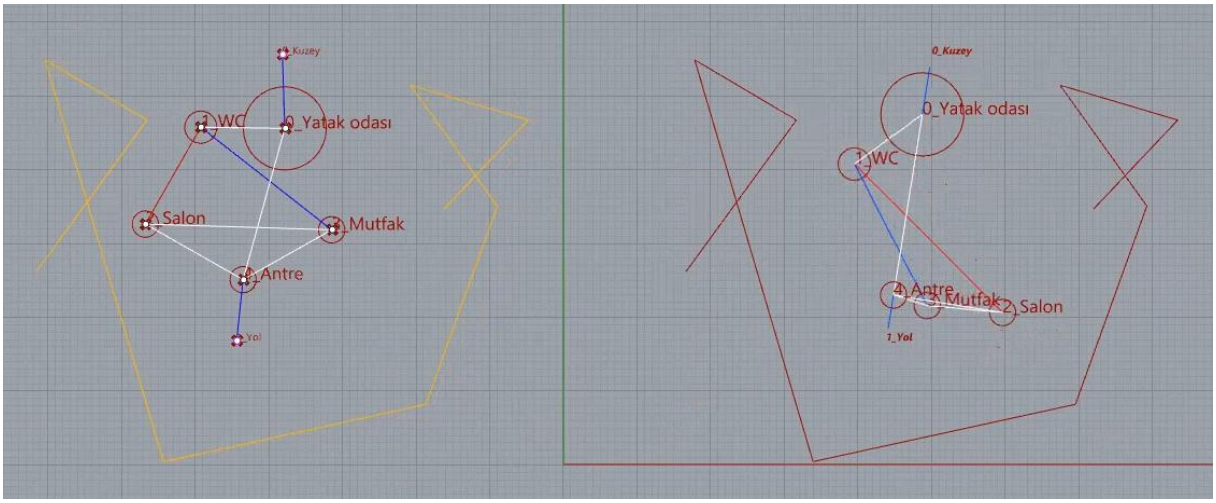
SpaceChase for Grasshopper ile üretilen dinamik ağların doğrudan kullanıcı etkileşimine açık diğer özellikleri ise fare yardımıyla gerçekleştirilen etkileşimler; ve ağ yapısı üzerinde gerçekleştirilen güncellemeler bağlamında ele alınabilir. Kangaroo altyapısının SpaceChase'e sağladığı önemli ve özgün bir yetenek, hareket esnasında düğüm noktalarının fare yardımıyla sürüklenmesine olanak sağlamasıdır (Şekil Ek 12-33).

Ağ yapısında yapılan değişiklikler, Rhinoceros ekranındaki nesnelerin (noktalar ve çizgiler) yerlerinin değiştirilmesi, isimlerinin değiştirilmesi, silinmeleri veya yeni nesnelerin eklenmesi biçiminde olabilir. Bu verinin *Geometry Pipeline* kullanılarak alındığı durumlarda SpaceChase yeni veriyi okuyarak harekete katmaya çalışacaktır. Bu tür değişikliklerin hareketli ağ yapısına sağlıklı biçimde aktarılabilmesi için *Bouncy Solver*'daki Reset düğmesine bir kere basmak gerekebilir.

Daha önce belirtildiği gibi, *Dynamic Graph Display* bileşeninin görselleştirdiği bağlantılar, sentaktik hesaplamalara dahil edilen ve edilmeyen tüm bağlantıları içermektedir. Bu bağlantıları ayırt etmek ve Grasshopper'da farklı biçimlerde görselleştirmek için bileşenin E çıktısından elde edilen verinin ayrıklaştırılması gerekmektedir. E çıktısındaki veri sıralı biçimde sunulmuştur ve bu sıralama, pozitif bağlantılar (sentaktik hesaplamalara dahil edilen ilişkiler), negatif bağlantılar ve bağlamsal bağlantılar (bağlamsal düğüm noktaları ile ilişkiler) biçimindedir. Bu veriyi ayıran bir Grasshopper kurulumu yapılarak bu bağlantıların farklı biçimlerde görselleştirilmeleri sağlanabilir. Bu ayrımın ardından bağlantılar ve düğüm noktaları *Custom Preview* veya *Text Tag 3D* gibi Grasshopper bileşenleri kullanılarak görselleştirilebilir. Bu kılavuzda gerçekleştirilen örnek çalışmanın son hali Şekil Ek 12-34'te sunulmuştur. Bu diyagram daha detaylı olarak Ek 10'da sunulmuştur. Şekil Ek 12-35'de bu çalışmanın sonucunda Rhinoceros ekranının durumu gösterilmiştir.



Şekil Ek 12-34. SpaceChase örnek dinamik ağ kurulumu. Bu diyagram daha detaylı olarak Ek 10'da sunulmuştur.

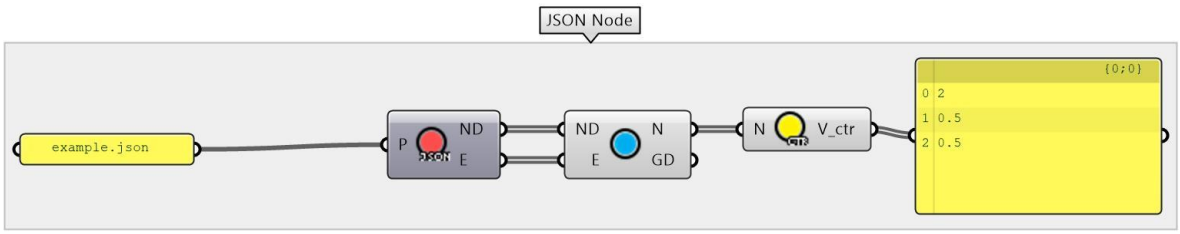


Şekil Ek 12-35. SpaceChase örnek dinamik ağının Rhinoceros ekranı. Sol tarafta kurulumda kullanılan geometrik girdiler, sağ tarafta ise SpaceChase tarafından harekete geçirilen ağ yapısı görülmektedir.

4. Veri Alış-Verişi

SpaceChase ile üretilen ağ yapılarının kaydedilebilmesi için hem Grasshopper'ın *.gh* uzantılı *definition* dosyasının, hem de Rhinoceros'ta oluşturulan nesneleri içeren *.3dm* uzantılı çizim dosyasının kaydedilmesi gerekmektedir.

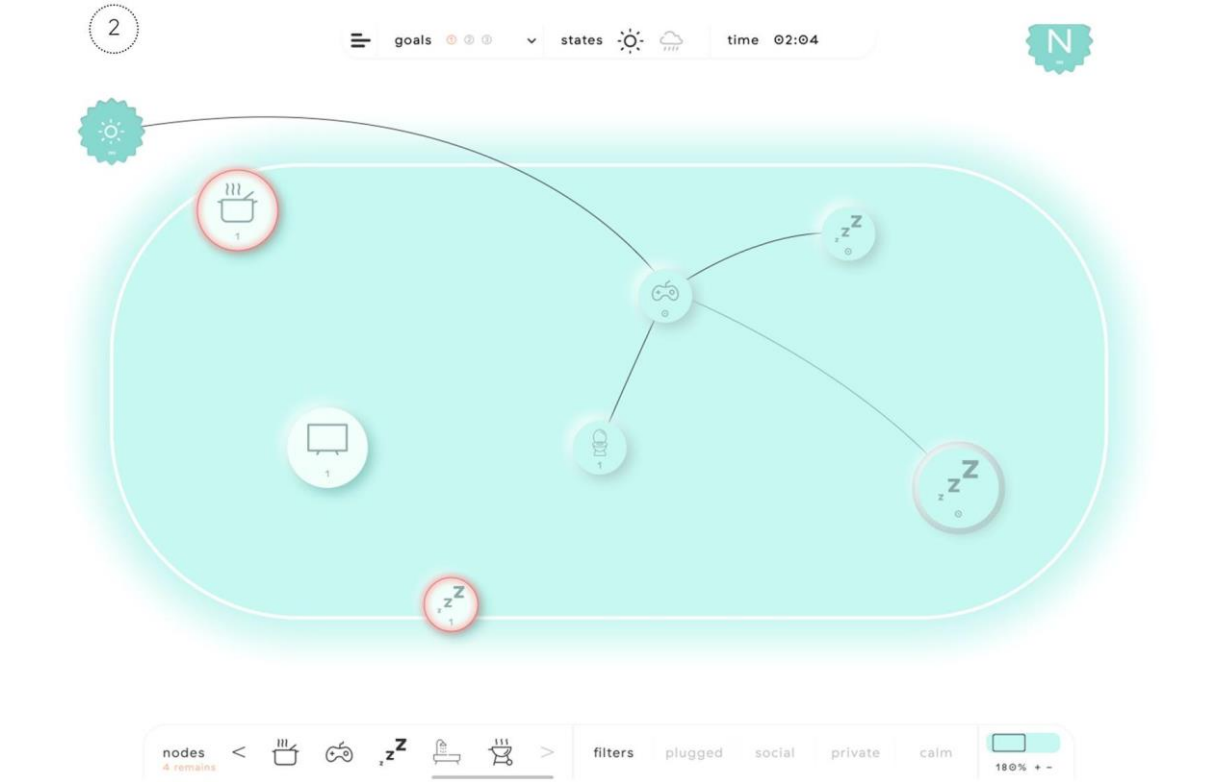
SpaceChase oyunundan çıkan ağ yapısını içeren *.json* formatındaki dosyalar SpaceChase içerisinde düğüm noktalarının ve bağlantıların türetilmesinde kullanılabilmektedir. .csv dosya türünde olduğu gibi, düğüm noktalarının üretildiği aşamada bu işlem için geliştirilmiş olan *Construct Json Node* bileşeni kullanılmalıdır. Şekil 36'da bu bileşenin kullanımı gösterilmiştir.

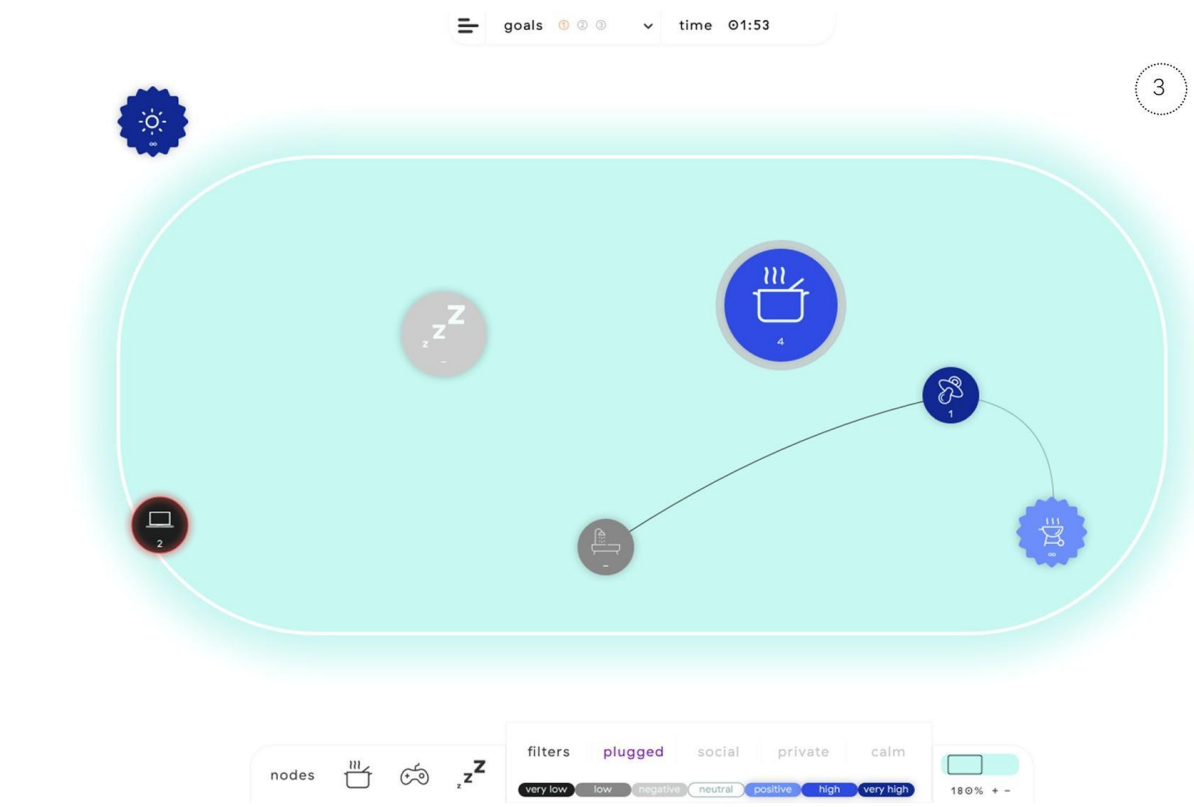


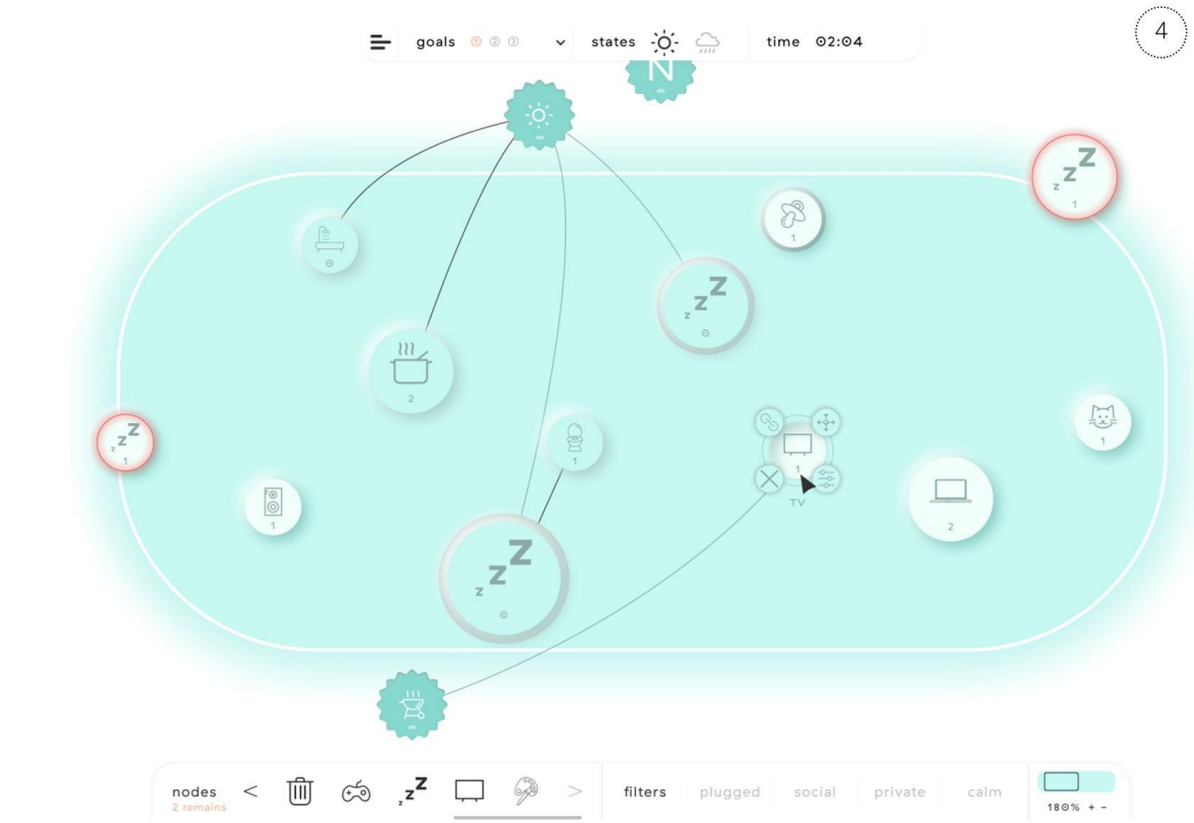
Şekil Ek 12-36. *Construct Json Node* bileşeni kullanılarak .json formatındaki ağ yapılarının SpaceChase içerisine aktarımı.

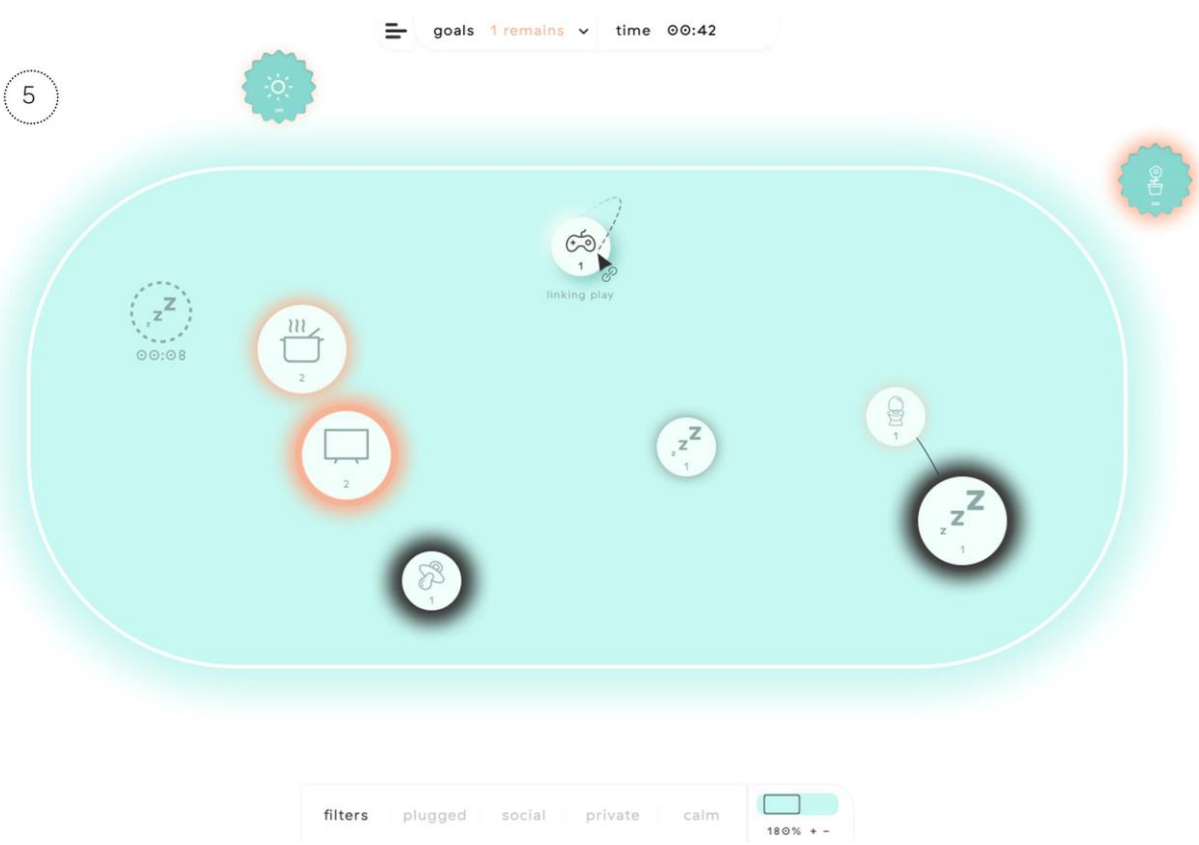
EK 13. SpaceChase Oyunu Arayüz Etkileşimini Gösteren Örnek Ekran Görüntüleri



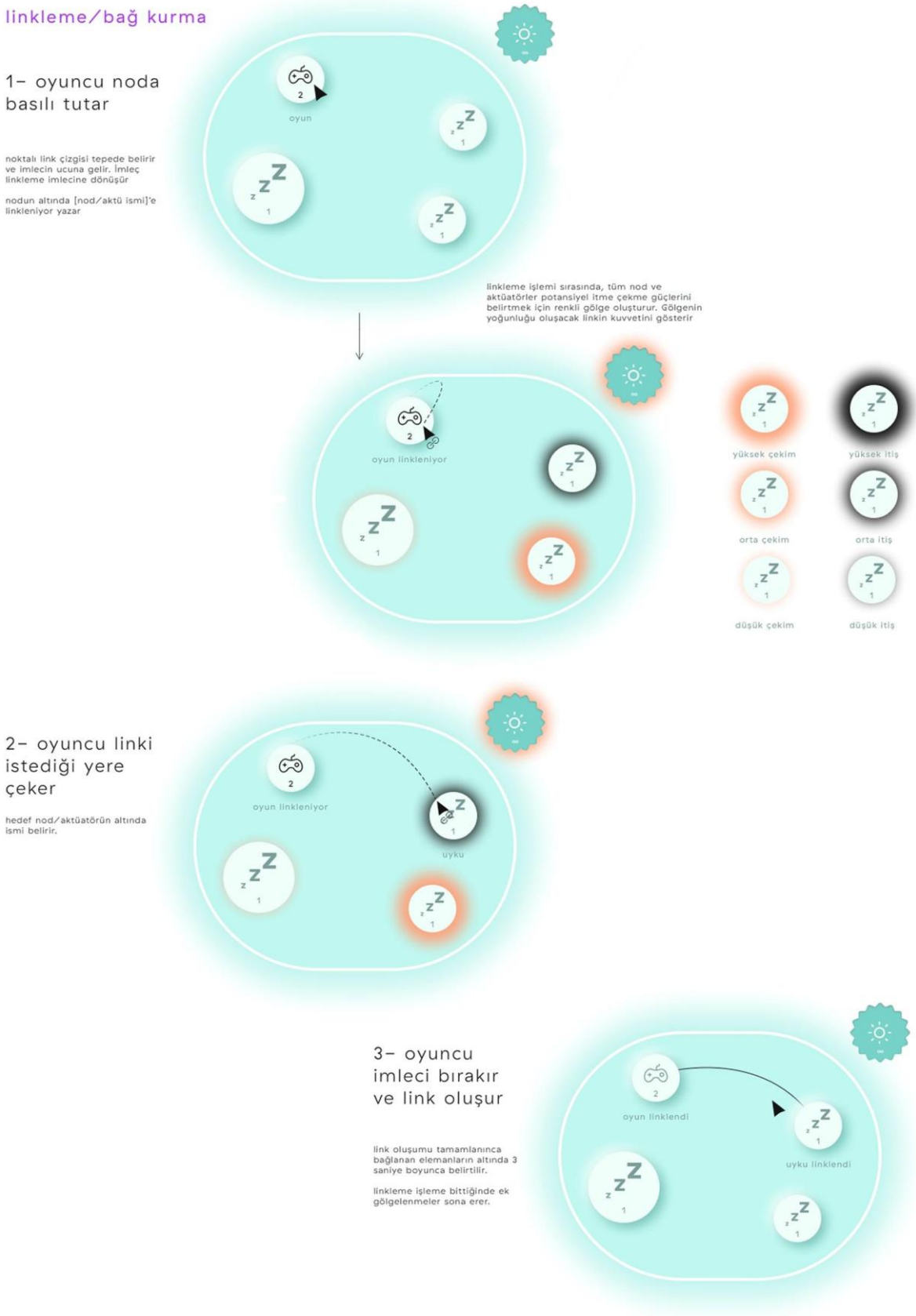



goals
time 01:53
3
nodes
filters
plugged
social
private
calm
very low
low
negative
neutral
positive
high
very high
180% + −


goals
states
time 02:04
4
nodes
2 remains
filters
plugged
social
private
calm
180% + −


goals 1 remains
time 00:42
5
00:08
linking play
filters
plugged
social
private
calm
180% + −

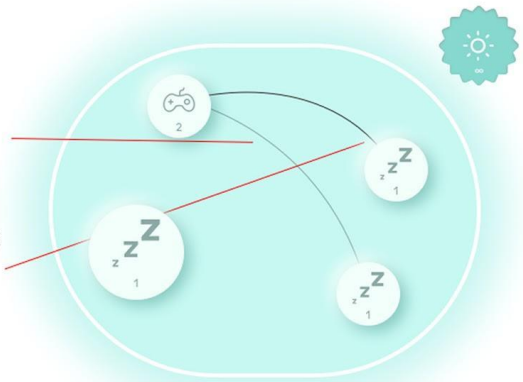
EK 14. Oyun Elemanları İlişkilerinin Anlatım Diyagramı



link renkleri

Çekim belirten link koyu yeşil

İtiş belirten link koyu gri



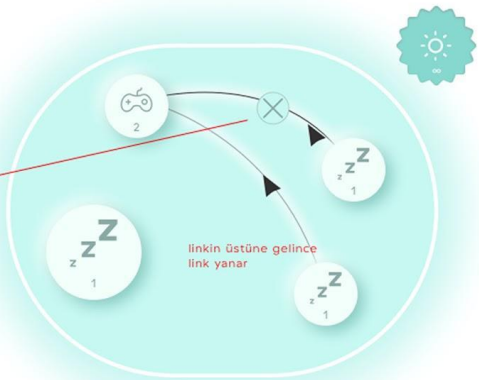
üzerine gelme ve yok etme

linkin üzerine gelindiğinde linkin 'ışığı yanar'.

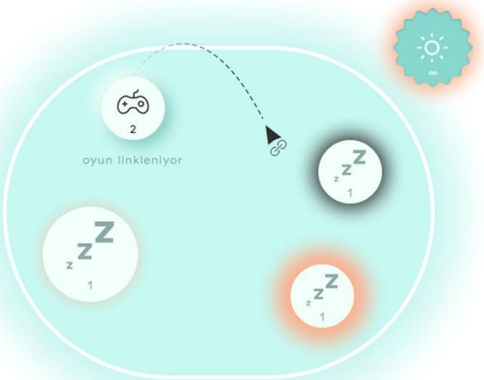
noda basılırsa nodu silme butonu linkin ortasında belirir. bu butona basıldığında onay gerekmeden link silinir.



silinen link 3 saniye içinde silinir.



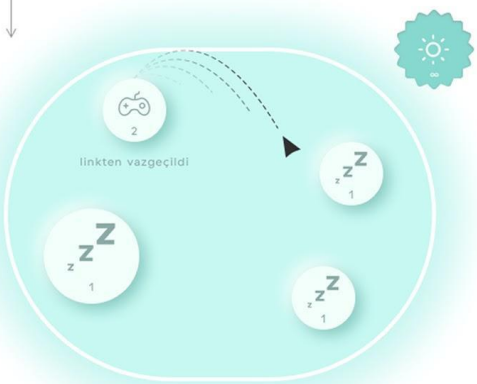
uyarılar:
Eğer oyuncu link kurarken parmağını boşlukta kaldırırsa, link çıktığı noktaya/etkin noda geri çekilir ve kaybolur.



bırakınca

linklenmeden vazgeçildi.

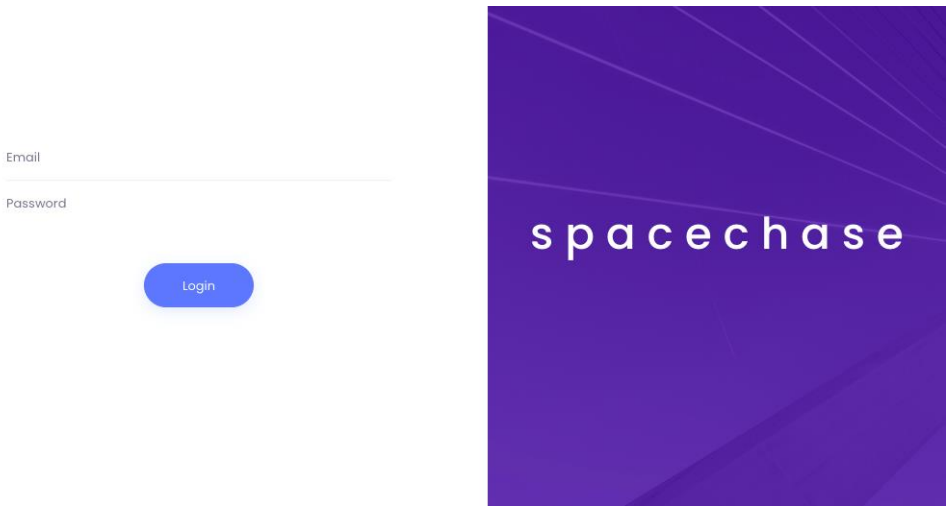
kesik çizgi başladığı yere dönüyor.



EK 15. SpaceChase Oyunu Kullanım Kılavuzu

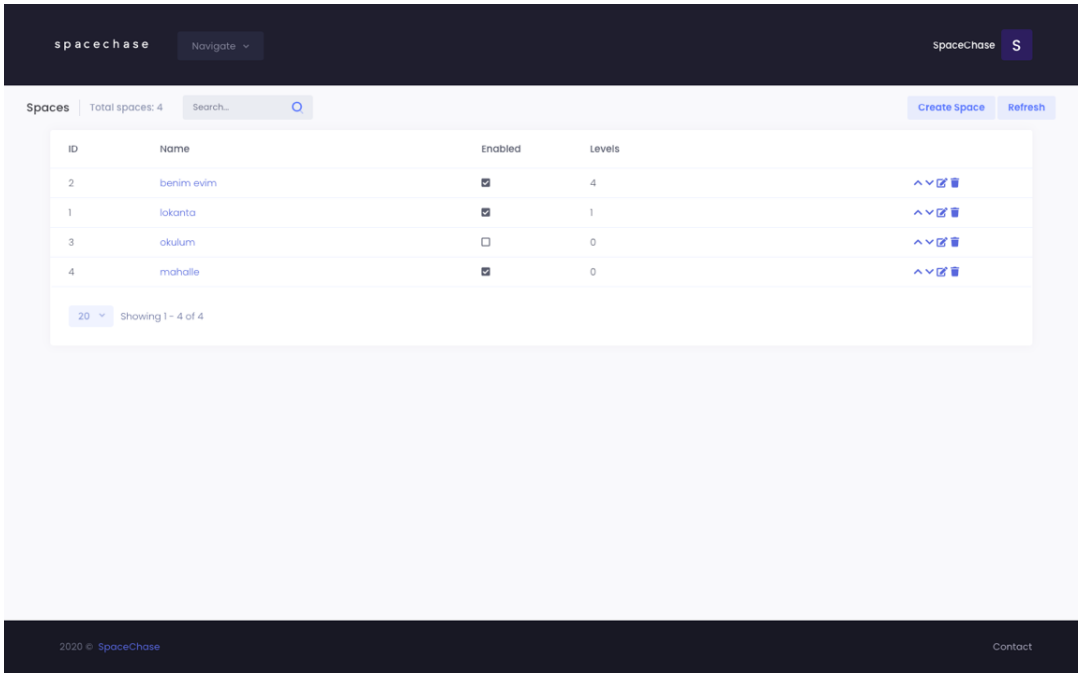
1. Yönetici Arayüzü İşleyiş Prensipleri

Yönetici Arayüzü, SpaceChase oyununda yer alacak sahneleri ve bölümleri tanımlamak amacıyla kullanılmaktadır. HTML tabanlı bir web yönetim arabirimi üzerinde tanımlanmakta olan Yönetici Arayüzü'ne https://admin.spacechase.app/ web adresi üzerinden erişilmektedir. Yalnızca oyun tasarım ekibi ve proje ekibinin erişimi olan bu adreste ilk olarak bir "giriş ekranı" ile karşılaşılır (Şekil Ek 15-1). Yönetici Arayüzünde, Mekanlar/Spaces", "Nodes/Düğümler", "Düğüm Etkileşimleri/Node Influences", "Uygulama/Application" ve "Oyuncular/Players" sekmeleri yer almaktadır.



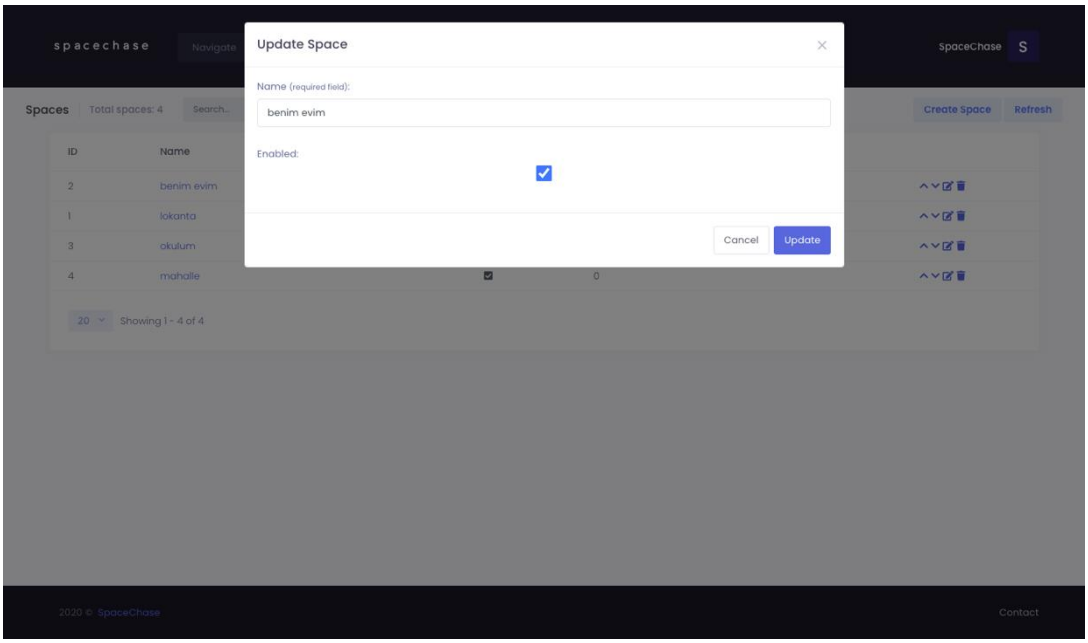
Şekil Ek 15-1. SpaceChase oyunu giriş ekran görüntüsü.

Sistem yöneticisi tarafından tanımlanan hesabının bilgilerini giren kullanıcı, "Mekanlar / Spaces" sekmesine yönlendirilerek mekan ve sahne gruplarının yer aldığı listeye erişim sağlar (Şekil Ek 15-2). Kullanıcı bu sekmede "mekan yarat/create space" butonuna tıklayarak yeni mekan çalışmaları üretebileceği gibi var olan mekan çalışmaları üzerinde de değişiklik yapabilmektedir. Kullanıcı, listenin sağ kolonunda yer alan yukarı ve aşağı ikonları aracılığıyla mekan sıralamasını değiştirebilmekte, kalem ikonu aracılığıyla mekan özelliklerini güncelleyebilmekte, çöp tenekesi ikonu ile de mekanı silebilmektedir.



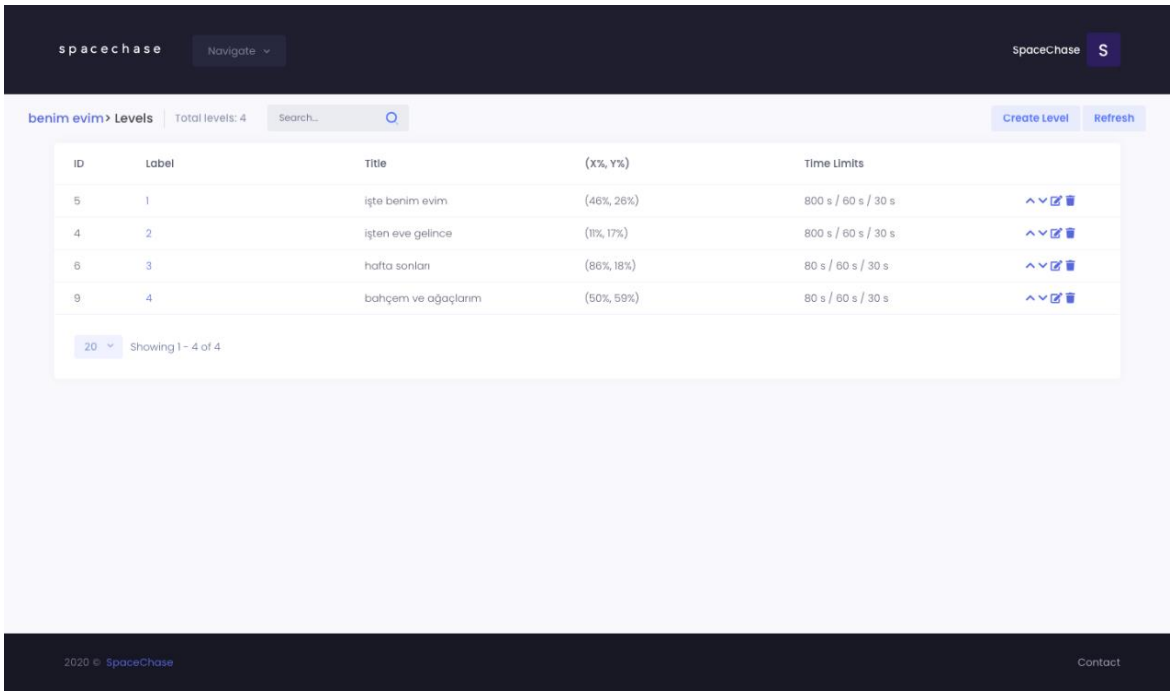
Şekil Ek 15-2. "Mekanlar / Spaces" sekmesi genel görünümü.

“Mekanlar/Space” sekmesinde yer alan sağ üst menüde ise, kullanıcının yeni bir mekan yaratması ve kullanıcı tarafından listelenen mekanların istenildiğinde güncellenebilmesi için fonksiyonlar sağlanmıştır. Kullanıcı, bu sekmede mekan ismini girebilmekte ve mekanın oyuncu menüsünde görülme durumunu seçebilmektedir (Şekil Ek 15-3).



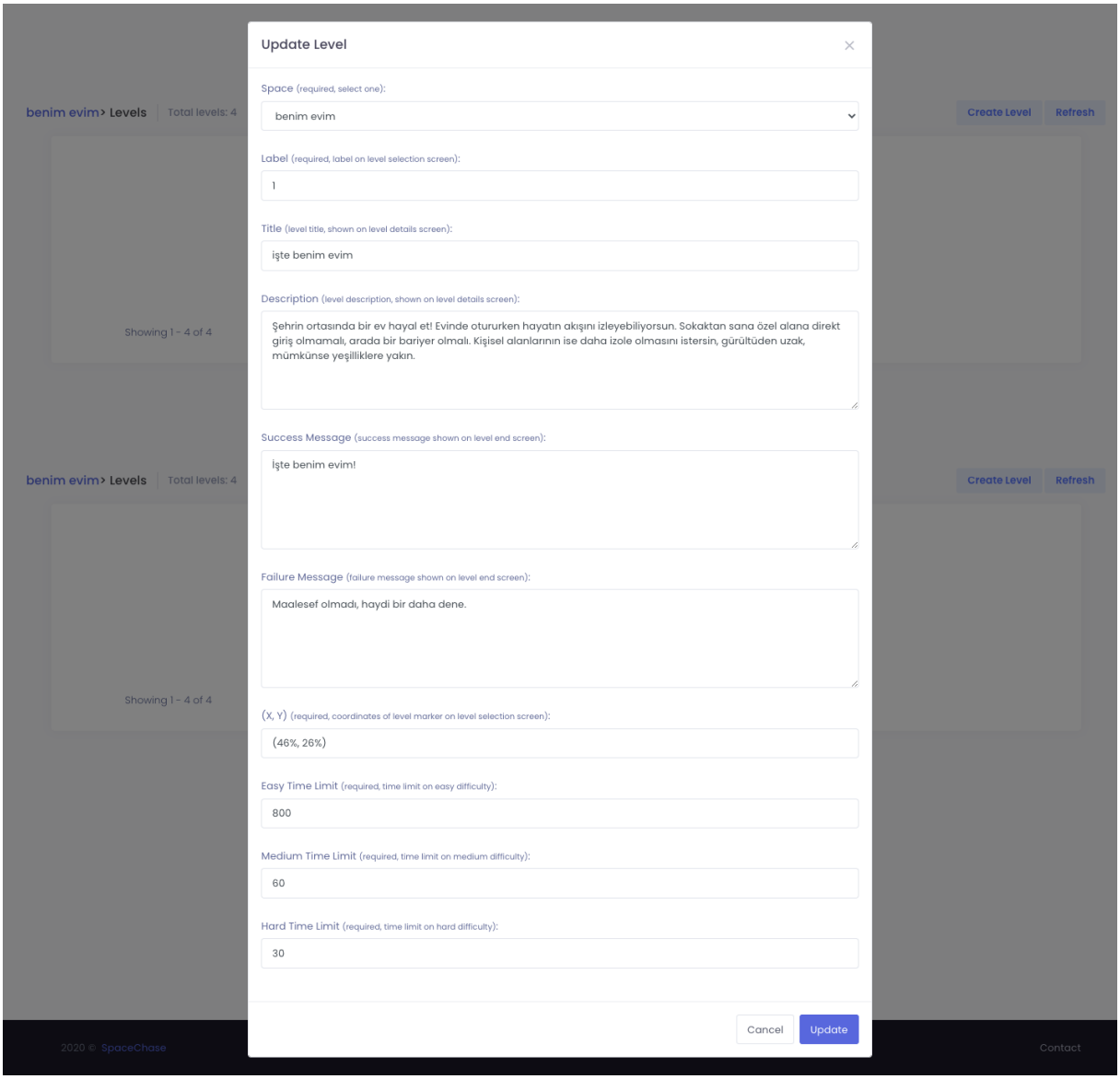
Şekil Ek 15-3. Mekan yarat *(create space)* sekmesi aracılığıyla mekanın öncül bilgileri girilir.

Yaratılmış bir mekanın ismine tıklayan kullanıcı, o mekan içinde yer alan bölümlerin bulunduğu listeye yönlendirilir. Bu listede, bahsi geçen bölümlerin adları, bu bölümlerin oyun ekranında hangi koordinatlarda gözükeceği ve bölümün zorluk seviyesine göre sahip olduğu zaman limitleri yer almaktadır (Şekil Ek 15-4). Kullanıcı, listenin sağ kolonunda yer alan yukarı ve aşağı ikonları aracılığıyla bölüm sıralamasını değiştirebilmekte, kalem ikonu aracılığıyla özelliklerini güncelleyebilmekte, çöp tenekesi ikonu ile de bölümü silebilmektedir. Sağ üst köşede yer alan "bölüm yarat/create level" menüsü ise seçili mekan içinde yeni bir bölüm yaratılması ve listelenen bölümlerin yenilenmesi için fonksiyonlar sağlamaktadır.



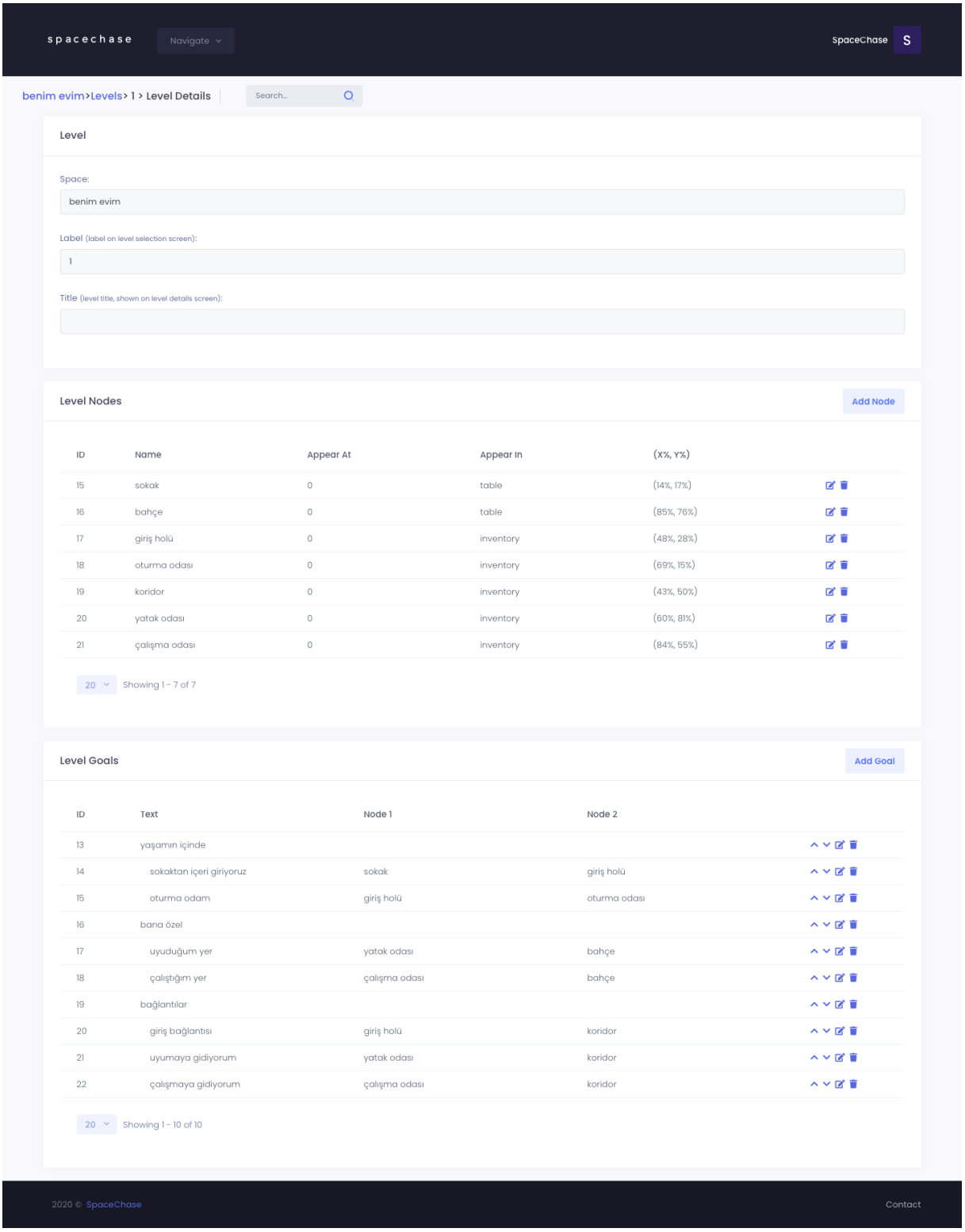
Şekil Ek 15-4. Sahne/bölümsekmesi içerik görüntüsü.

Yaratılmış bölümün ismine tıklayan kullanıcı oyun özelliklerinin yer aldığı sekmeye yönlendirilir. Bu sekmede bölümü meydana getirmek için şu adımlar takip edilmiştir; (1) "space/mekan" bölümünde bölümün yer alacağı sahne/mekan seçilir, (2) "label/etiket" bölümüne o bölümün seviyesi yazılır, (3) "title/başlık" kısmında bölümün sahip olduğu başlık yazılır, (4) "description/tanım" kısmında oyuncuya bölüme ilişkin bilgi vermek amacıyla tanımı yapılır", (5) "success message/başarı mesajı" başlığına oyuncunun hedeflenen adımları gerçekleştirerek oyunu tamamlaması durumunda karşılaşacağı metin yazılır, (6) "failure message" başlığında oyuncunun hedeflenen adımları gerçekleştirememesi durumunda oyun sonunda karşılacağı metin yazılır, (7) (X,Y) başlığına bölümün oyun ekranında hangi koordinatlarda gözükeceği yazılır, (8) bölümün zorluk seviyesine göre sahip olduğu zaman limitleri yazılır. Bu kapsamda kullanıcı tariflenen adımlarını tamamlayarak mekan için gerekli senaryoyu meydana getirmiş olur (Şekil Ek 15-5).

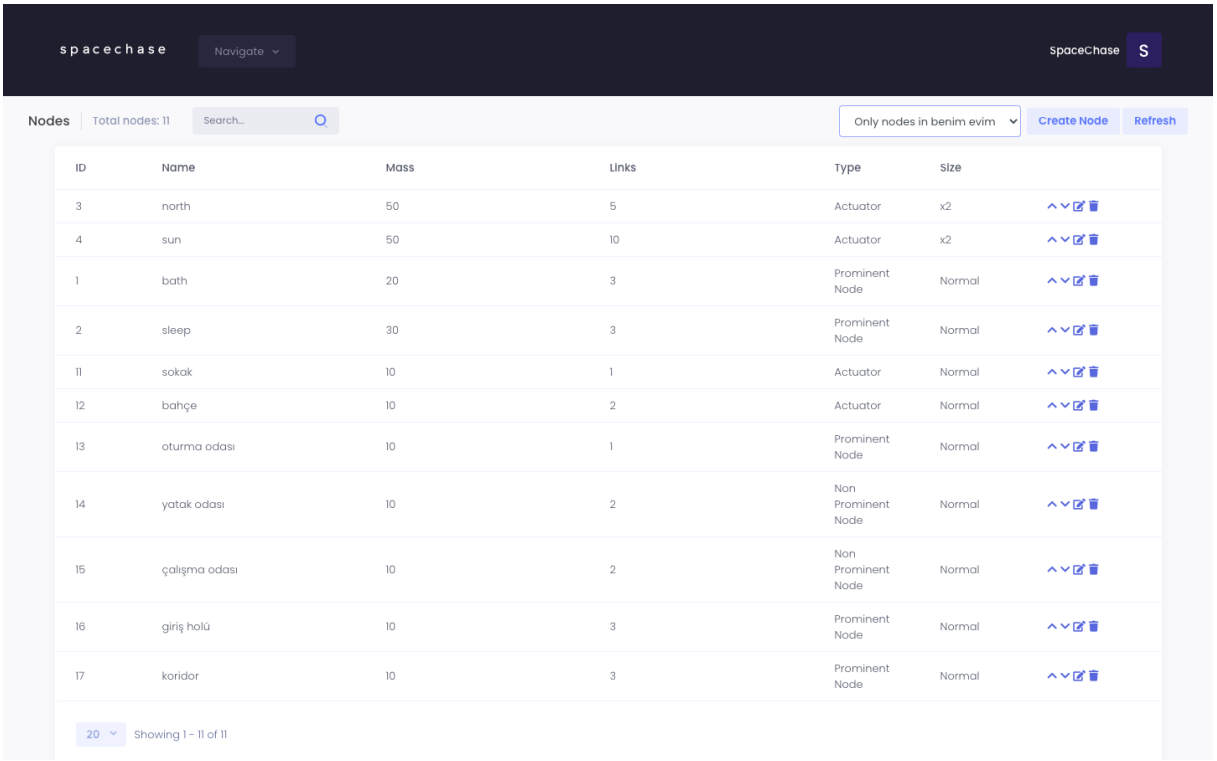


Şekil Ek 15-5. Sahneyi meydana getirmek için gerekli adımları oluşturan başlıklar.

Oyuncunun oluşturulan senaryo tanımı dahilinde mevcut bulunan düğüm noktaları arasında ilişki kurması beklenmektedir. Kullanıcı bu ilişkileri tanımlamak için “level/bölüm” etiketinin üstüne tıklayarak bölüm detay ekranına yönlendirilir (Şekil Ek 15-6). Bölüm detay ekranında yer alan “level nodes” listesinde bölüm tanımı içindeki olguların oyun içerisinde gösterilebilmesi için çeşitli düğüm noktaları tanımlanırken; “level goals” listesinde oyuncunun bölümü başarılı bir şekilde tamamlaması için gereken ilişkiler tariflenmektedir.

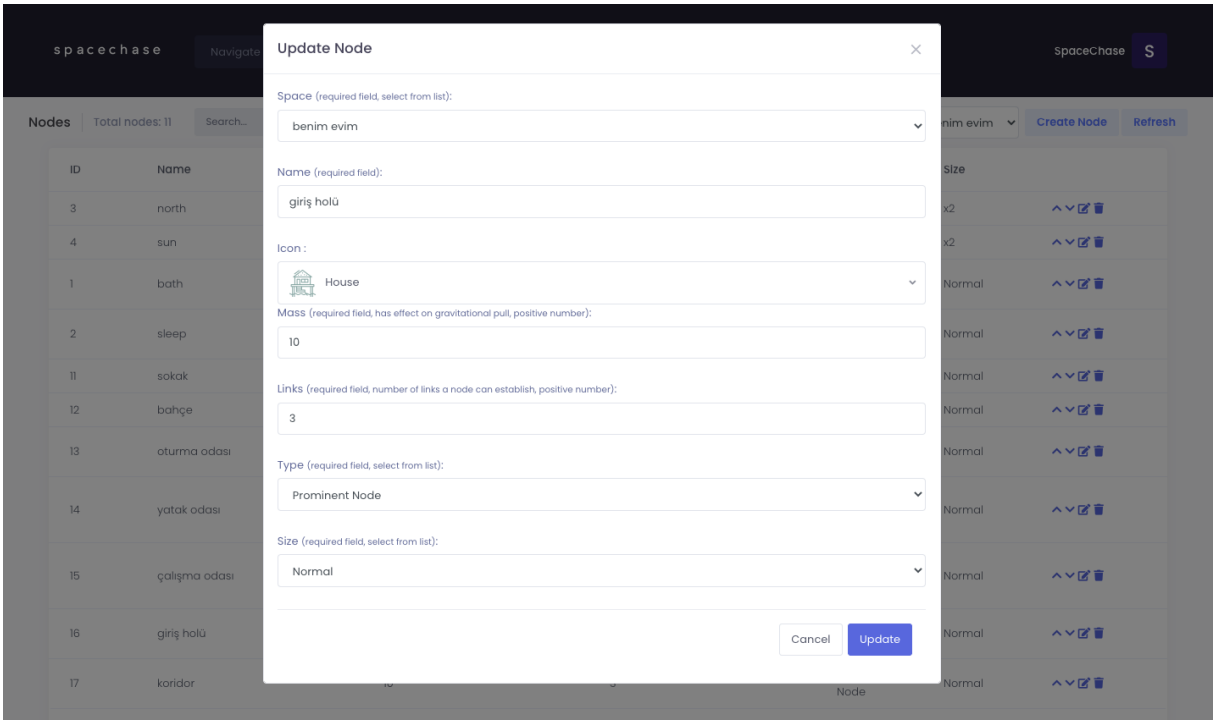


Şekil Ek 15-6. Bölüm detay ekranı içerik başlıkları.



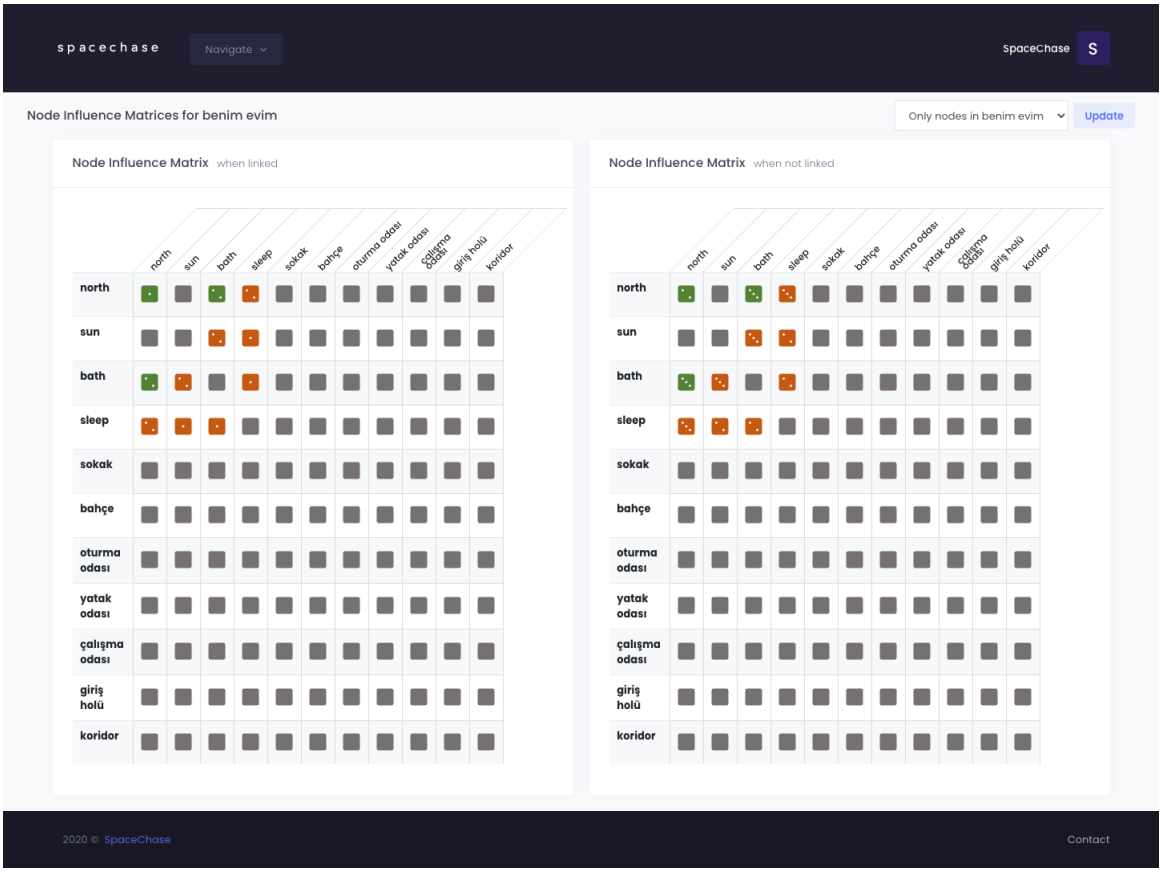
Şekil Ek 15-7. Düğüm noktası liste sekmesi.

Düğüm noktası/Nodes listesinde düğüm noktasının adı, hangi mekanda kullanıldığı ve ikonu tariflenmektedir (Şekil Ek 15-7 ve 15-8). Oyun ekranında gerçekleşen simülasyonda kullanılmak üzere düğüm noktasının kütlesi, oluşturabildiği bağlantı sayısı, boyutu ve tipi tariflenmektedir. Düğüm noktaları, “bağlamsal düğüm”, “öncelikli” düğüm ve “düşük öncelikli” düğüm olmak üzere üç farklı tipte oyunda yer almaktadır. "Bağlamsal" düğümler, oyuncu tarafından hareket ettirilemeyerek sahnenin/bölümün sabit parçalarını oluşturmaktadır. “Öncelikli” düğümler ve “düşük” öncelikli düğümler “bağlamsal” düğümlerin aksine sürükle bırak yöntemi ile hareket ettirilebilmektedirler. "Öncelikli" düğümler, sadece oyun ekranı sınırları dahilinde hareket edebilirken “düşük öncelikli” düğümler oyun ekranı dışına çıkma eğilimi göstermektedirler. “düşük öncelikli” düğümler oyun ekranından çıktıkları takdirde ekrandan kaybolur, oyuncu bölümü yeniden başlatana kadar da oyun ekranında yeniden belirmezler.



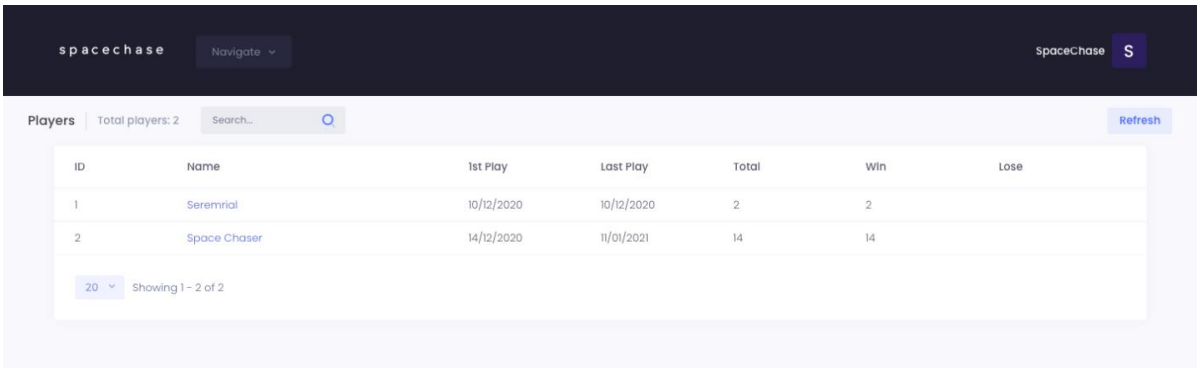
Şekil Ek 15-8. Düğüm noktası özelleştirilmiş içerik paneli.

Düğüm noktaları oyuncunun müdahalesi dışında Newton fiziğini prensip olarak kullanarak da hareket edebilmektedir. Evrensel kütle çekim yasası ile düğüm noktaları arasındaki anlık kuvvetler hesaplamaktadır. Bu kuvvetler ve düğüm noktaları arasındaki etkileşim katsayıları kullanılarak birbirleri üzerinde etki gösterirler ve noktaları hareket ettirirler. Bu etkiler, yeşil renkle ifade edilen pozitif çarpanlar ile çekme kuvveti; kırmızı ile ifade edilen negatif çarpanlarla ise itme kuvveti yaratır (Şekil Ek 15-9). Gri sembol ile tarif edilen sıfır çarpanına sahip düğüm noktaları arasında etkileşim olmamaktadır. Bu çarpanlar, oyuncunun noktalar arasında ilişki kurup kurmamasına bağlı olarak farklı değerlere sahip olabilmektedir. Tüm bu değerler kullanılarak oyun içi simülasyonlar gerçekleşmektedir.

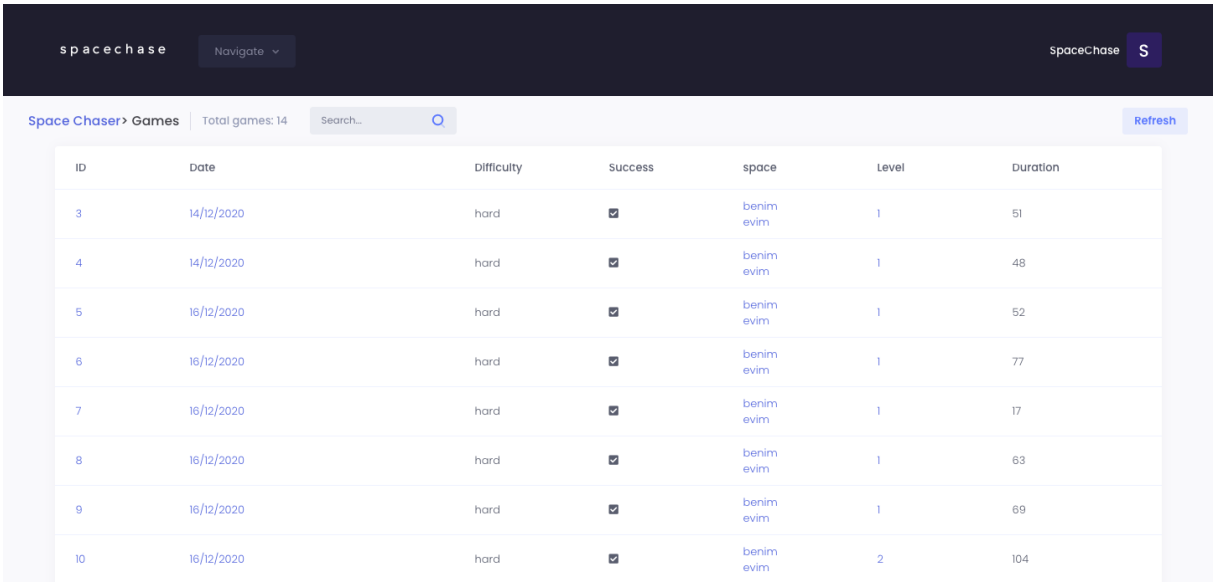


Şekil Ek 15-9. Düğüm noktaları arasında oluşturulan ilişki matrisi.

Bölümler, oynandıktan sonra oluşan grafikler incelenebilmektedir. Oyuncular kullanıdıkları platformlara göre otomatik olarak gruplanmakta ve listelenmektedirler. Şekil Ek 15-10'da yer alan listede oyuncunun oynadığı tüm bölümlere erişilebilirken Şekil Ek 15-11'de yer alan listede oynanan bölümün sonucunda oluşan grafiğe erişilmektedir.

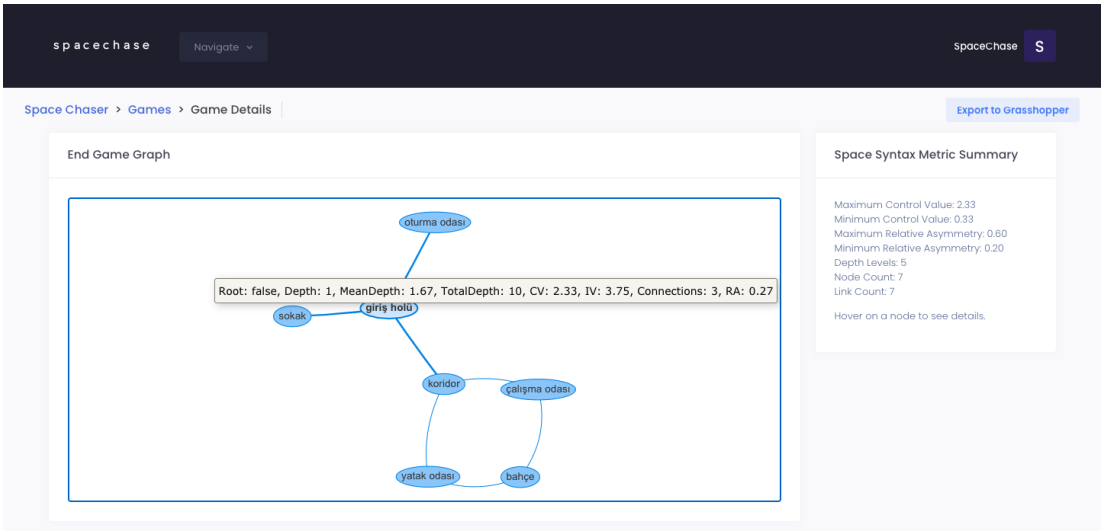


Şekil Ek 15-10. Oyuncunun oynadığı tüm bölümlerin listelenmesi.



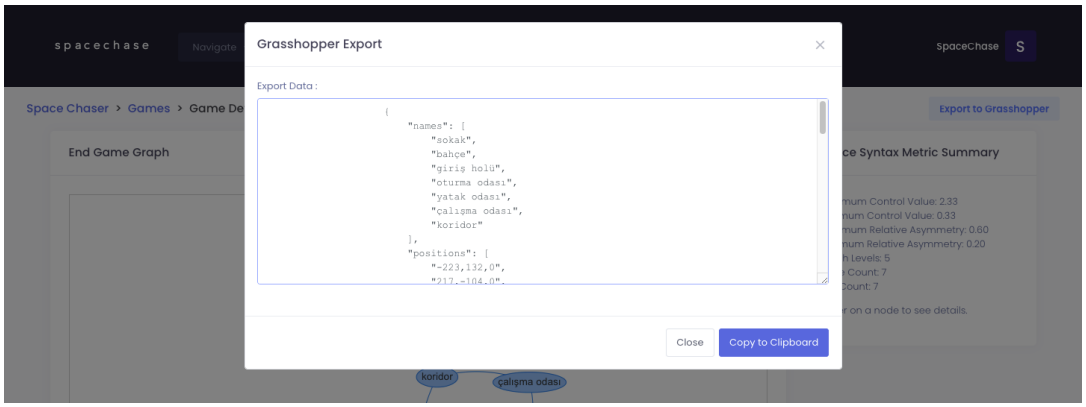
Şekil Ek 15-11. Oynanan bölüm sonucunda oluşan grafiğin listelenmesi.

Şekil Ek 15-12'de yer alan ekranda grafiğin mekan dizimsel metrikleri hesaplanıp gösterilmektedir. Minimum ve maksimum kontrol değerleri, minimum ve maksimum entegrasyon değerlerinin yanında, grafikteki düğüm noktası sayısı, bağlantı ve derinlik değerleri, ekranın sağ tarafındaki panelde gösterilmektedir. Bunun dışında, ekrandaki grafiğin herhangi bir noktası üzerine gelindiğinde, o noktanın geçiş grafiğinin başlangıç noktası olup olmadığı, derinlik, kontrol değeri, entegrasyon değeri ve kurduğu bağlantı sayısı da gösterilmektedir.



Şekil Ek 15-12. Oyun sonucunda meydana gelen grafiğin Mekan Dizimi metrikleri ile analiz edilmesi.

Oluşan bu grafiği Grasshopper uygulaması ile daha detaylı bir şekilde analiz etmek isteyen araştırmacı için, SpaceChase for Grasshopper ile uyumlu olacak şekilde aktarma özelliği sunulmuştur (Şekil Ek 15-13).



Şekil Ek 15-13. SpaceChase oyununda elde edilen sonuç SpaceChase for Grasshopper uygulamasına aktarılarak veri geçişine imkan vermektedir.

2. Oyuncu Arayüzü İşleyiş Prensipleri

Oyuncu, https://play.spacechase.app/ bağlantısı aracılığıyla Oyun Arayüzü'ne ulaşarak "ana menü" ekranına yönlendirilmektedir. Bu menüden "play" butonuna basarak sahnelerin listesinin görüntülendiği ekrana geçilmektedir (Şekil Ek 15-14).



Şekil Ek 15-14. Oyuncu arayüzü giriş paneli.

Sahne listesi ekranında, sağ alt tarafta, oyun için tanımlanmış sahnelerin/mekanların listesi gelmekte ve mekanlar arası geçiş yapılabilmektedir. Seçili mekanın içinde tanımlanmış bölümler ise numaralandırılmış daireler şeklinde ekranda yer almaktadır (Şekil Ek 15-15).



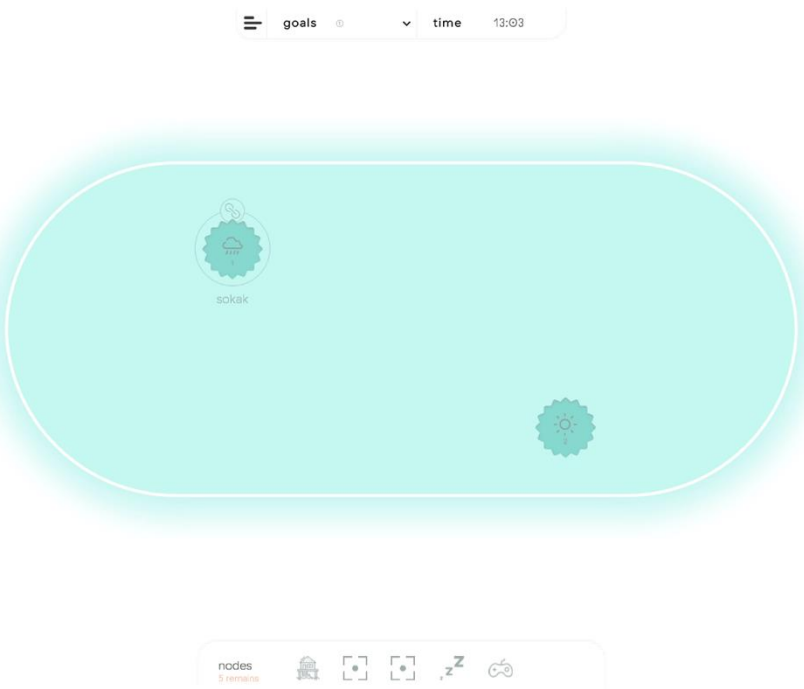
Şekil Ek 15-15. Sahne/bölüm listesi genel görünüm düzeni.

Oyuncu, tanımlanan bölüm ikonlarına tıklandığı zaman bölüm detaylarını görmek üzere yönlendirilir (Şekil Ek 15-16). Oyuncunun yönlendirildiği ekranda, “about” kısmında oyuncunun seçmiş olduğu bölüme dair açıklama yer alırken “goals” kısmında oyuncunun ulaşması beklenen hedefler gösterilmektedir. Oyuncu oynamak istediği zorluk seviyesini “difficulty” kısmında seçtikten sonra, "play" butonuna basarak bölümü başlatır.



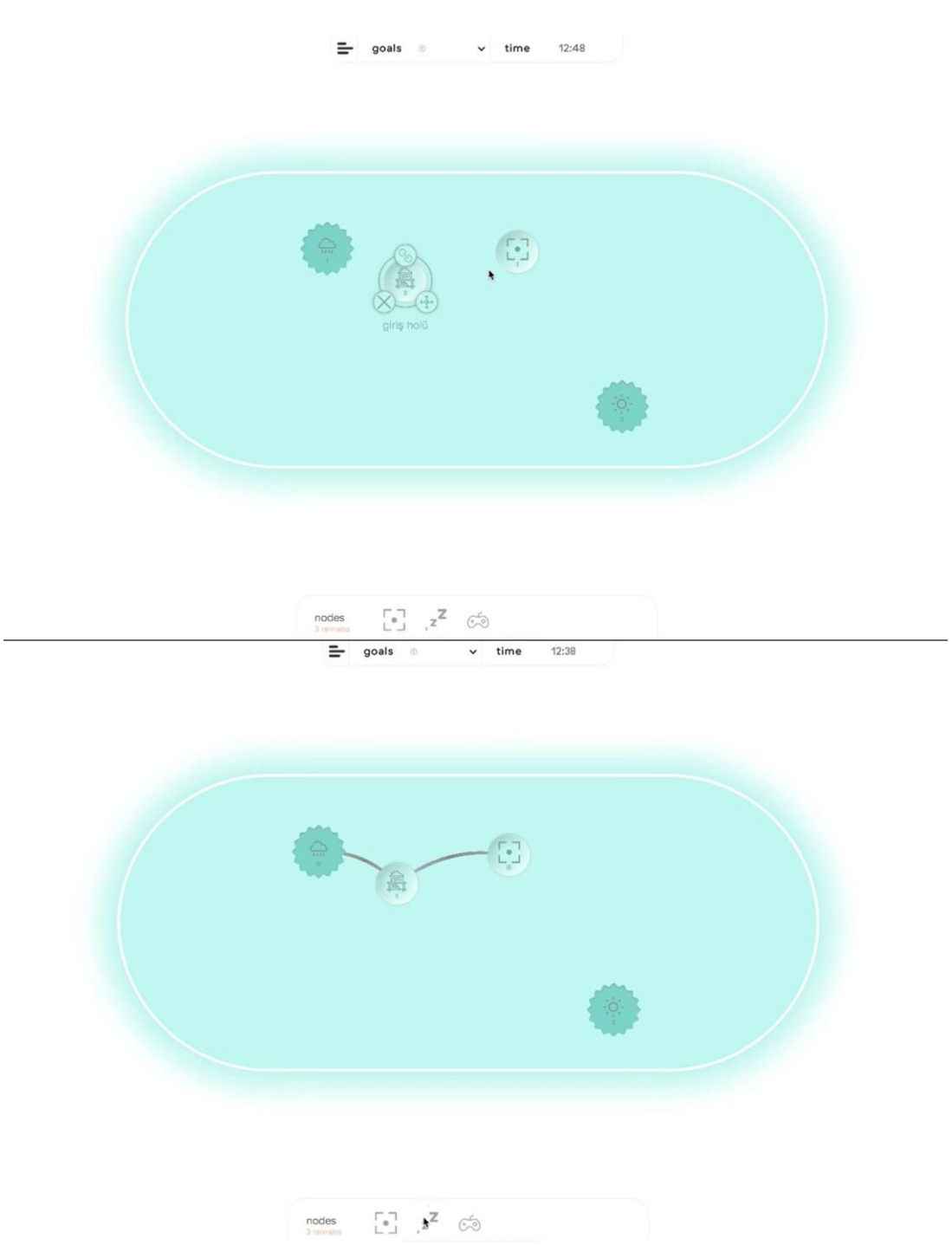
Şekil Ek 15-16. Bölümdetay paneli genel görünümü.

Bölüm başladığında oyuncunun yerini değiştiremediği bağlamsal düğüm noktaları olarak tarif edilen bileşenler, oyun ekranında yer alır. Oyuncu, Şekil Ek 15-17'de oyun ekranındaki alt barda görünen düğüm noktası listesinden seçim yaparak diğer düğüm noktalarını oyun alanına taşıyabilmektedir.

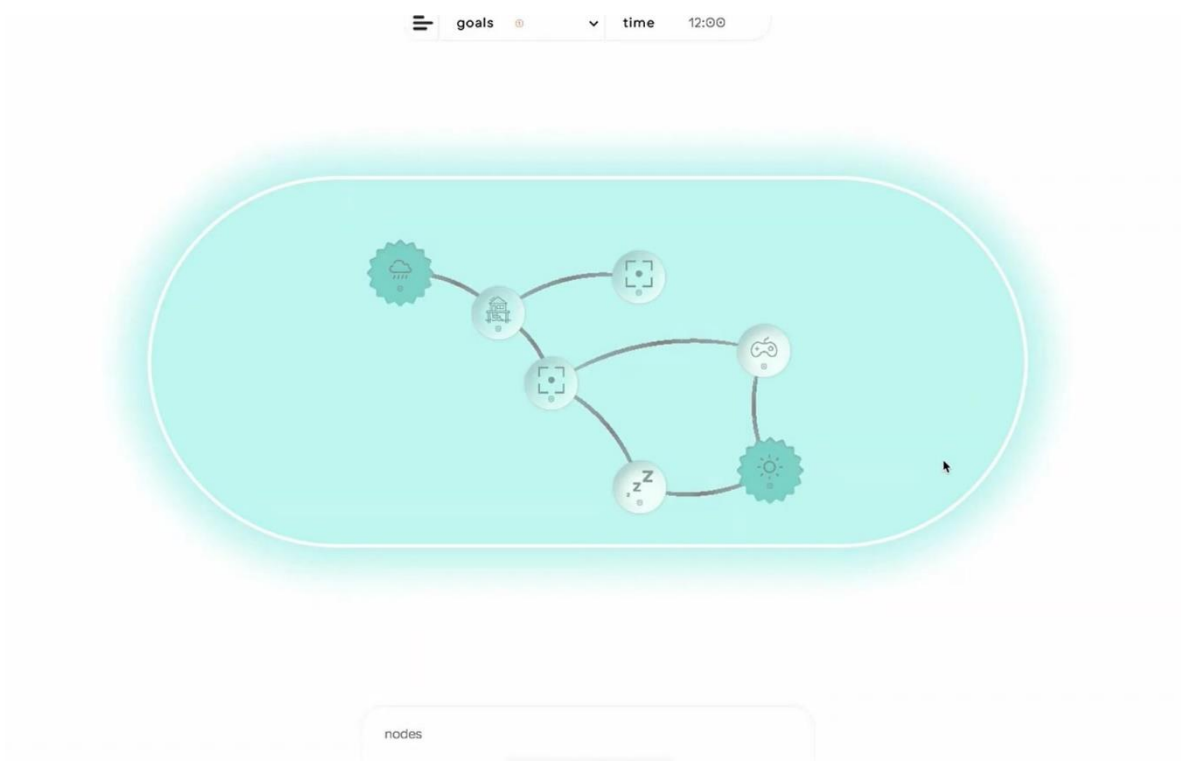


Şekil Ek 15-17. Düğüm noktalarının oyun ekranındaki ilişkileri.

Oyuncu, bölüm gereği oyunda yer alan farklı düğüm noktalarını ekrana taşımakta, düğüm noktalarının üzerine tıklayıp imleci hareket ettirerek bu noktalar arasında ilişkileri kurmaktadır (Şekil Ek 15-18). Şekil Ek 15-19'da görüldüğü gibi bölüm tarifinde belirtilen hedefler doğrultusunda tüm ilişkileri doğru olarak kuran oyuncu bölümü başarılı olarak tamamlamış olur (Şekil Ek 15-20).



Şekil Ek 15-18. Düğüm noktaları arasında kurulan bağın işleyiş süreci.



Şekil Ek 15-19. Oyun ekranında bağların ve düğümlerin olası ilişkilerinin görünümü.



Şekil Ek 15-20. Bölüm sonunda görünen başarı ekranı örneği.

EK 16. Atölye I Programı

SPACECHASE FOR GRASSHOPPER_WIP // WORKSHOP I

- **İTÜ Taşkışla | 20 öğrenci** (2 kişilik gruplar)
- **5 Aralık Perşembe** 09.30-12.30 (3 Saat)
- **12 Aralık Perşembe** 09.30-12.30 (3 Saat)

Birinci gün (05.12.2019)

- Giriş / Tanıtım / Seminer (Pelin Dursun Çebi)
- Konut şeması üzerinden **ikide1** oyununun oynanması. *(hazırlık – oyun – tartışma)*
- **Cytoscape** ve mekansal konfigürasonlarla ilişkisinin anlatımı (Büşra Balaban + Nilüfer Kozikoğlu)
- Oyuna dair ilişki matris ve listelerinin oluşturulması.
- Cytoscape ile görsellerin ve sayısal verinin elde edilmesi.
- Her grubun (2 kişi) bir kreş, bir konut, bir restoran örneği seçmesi ve bu yapıları bir sonraki haftaya matris ve listelerini oluşturup Cytoscape'te modelini almış olarak gelmesi beklenir.

* Gün boyu var olan uygulamaları kullanılırken katılımcılar gözlemlenip kayıt edilecek. Oyun oynanırken video kaydı alınacak.

İkinci Gün (12.12.2019)

- Syntactic programının anlatımı ve birlikte deneyimlenmesi (Oğulcan Üneşi)
- Seminer (Nilüfer Kozikoğlu)
- SpaceChase anlatımı, deneyimlenmesi ve kullanıcı kayıtlarının alınması.
- Tartışma, değerlendirme.
- Öğrencilerden kullandıkları üç ayrı programın olumlu/olumsuz yönlerine dair eleştirilerini yazmaları ve bir sonraki hafta teslim etmeleri istenir.

* Gün boyu var olan uygulamaları kullanırlarken katılımcılar gözlemlenip kayıt edilecek. Oyun oynanırken video kaydı alınacak.

EK 17. Atölye II Programı

SPACECHASE FOR GRASSHOPPER_WIP // WORKSHOP II

- **Çevrimiçi Atölye | 13 katılımcı**
- **03.06.2020** 11.00-14.00 (3 Saat)
- **04.06.2020** 11.00-14.00 (3 Saat)

Birinci gün (03.06.2020)

• 11:00 Seminer (Pelin Dursun Çebi)

• 11:30 SpaceChase anlatımı (Oğulcan Üneşi+Tuğrul Yazar)

• 12:00 **Egzersiz 1:** Peter Eisenmann House VI konut şemasını öğrendiğiniz aracı kullanarak modelleyin...

• 12:30 Grasshopper State kaydedip, gh + 3dm teslimi

• 12:30-13:00 Öğle arası

• 13:00 **Egzersiz 2:** Size verilen konut şemasını modelleyin. Her kullanıcıya farklı bir konut şeması verilerek 5-10 dakika gibi bir süre içerisinde canlı yayında modellemesi istenecek ve kayıt alınacak.

• 13:45-14:00 **Anket:**

• > Size verilen görevde kullandığınız aracın ne kadar faydalı olacağını düşünüyorsunuz?

• > Bugün kullandığınız araç ve yöntemle ilgili üç anahtar kelime yazınız.

• > Kendinizi verilen görevlerde ne kadar başarılı buldunuz?

• > Bu aracın mesleki olarak sizi ne kadar etkileyeceğini düşünüyorsunuz?

İkinci gün (04.06.2020)

• 11:00 Seminer (Nilüfer Kozikoğlu)

• 11:30 **Egzersiz 3:** Öğrendiğiniz aracı kullanarak **iki katlı müstakil konut şeması** tasarlayın...

• 12:00 Grasshopper Stage kaydederek gh + 3dm teslimi alınacak.

• 12:00 **Sunumlar:** Her öğrencinin kendi konut şemasını göstererek yapmak istediklerini/ yapabildiklerini sunarak anlatması / göstermesi kaydedilecek.

• 12:30-13:00 Öğle arası

• 13:00-13:15 **Anket:**

• > Size verilen görevde kullandığınız aracın ne kadar faydalı olacağını düşünüyorsunuz?

• > Kendinizi görevde ne kadar başarılı buldunuz?

• > Bu aracın mesleki olarak sizi ne kadar etkileyeceğini düşünüyorsunuz?

• 13:15-14:00 Görüşme

EK 18. Atölye III Programı

SPACECHASE FOR GRASSHOPPER_WIP // WORKSHOP III

- **Dynamic Canvas | Online Workshop | 10 participants**
- **14 September 2020** 10.00 – 14.00
- **15 September 2020** 10.00 – 14.00

Dynamic Canvas Workshop is based on *SpaceChase*, game-like syntactic interface research for architectural design process based on network thinking and Space Syntax. The tool enables the participants to play and adjust relationships between the variables of a project, mainly spatial and contextual in a very loose metric analogy and access this information to arrive at design decisions as well as evaluate them.

The workshop program includes a seminar on space syntax, network thinking, graph theory applications and design probing, followed by introduction of the design interface. Participants work on a rubric that is derived out of existing projects organizational layout data and reinterpretations, and iterative design sessions with comparative presentations.

The aim of the interface is not specifically to arrive at planning the configurational planimetric data, but to enable the architectural sketch session that used to be within the hand/paper realm within the coded computerized environment augmented with a series of evocative chance inclusive visualization potentials. The user is able to conceptualize the relationships between the design elements contextual or introduced programs, revise, add, amplify or break on the go as well as possible new relationships that are advised by the preloaded library of projects.

* SpaceChase is an academic design research supported by the ARDEB program of TUBITAK governmental academic research program. Dynamic Canvas is a term coined by Nilufer Kozikoglu to describe the ever-changing mode of the design space, i.e. design criteria, elements and the context.

- Focus

Workshop is an opportunity to share current Grasshopper plugin with the participants, and discuss the potentials the tool provides for creative as well as efficient design process. The participants will be guided on how to use the software in the initial day and asked to use it to re-program a space syntax graph of and existing layout, or program one out of scratch. The participants will be asked to contribute to assess the process in the final session.

- Requirements

Rhinoceros V6 and Grasshopper.

- **Workshop Program (4 hours x 2 days)**

Day 1 (14.09.2020)

- 10:00-11:00 Introduction and Seminar (Nilüfer Kozikoğlu, Pelin Dursun Cebi)
- 11:00-12:00 SpaceChase for Grasshopper Tutorial (Can Üneşi, Tuğrul Yazar)
- 12:00-13:00 Exercise:
- 13:00-14:00 Exercise:
- 14:00-14:15 Online survey
- Discussion-Moderated by Nilufer Kozikoğlu

Day 2 (15.09.2020)

- 10:00-11:00 Seminar (Nilüfer Kozikoğlu)
- 11:00-12:00 Exercise:
- 12:00-13:00 Discussion:
- 13:00-14:00 Exercise:
- 14:00-14:15 Online survey
- Evaluation

EK 19. Atölye IV-V Programı

SPACECHASE FOR GRASSHOPPER_WIP // WORKSHOP IV-V

- **Çevrimiçi Atölye | 13 öğrenci**
- **19/11/2020** 09.30-12.30 (3 Saat)
- **26/11/2020** 09.30-12.30 (3 Saat)

Birinci gün (19.11.2020) ATÖLYE IV

- 09:30 Seminer (Pelin Dursun Çebi)
- 09:45 SpaceChase for Grasshopper anlatımı (Can Üneşi – Tuğrul Yazar)
- Senaryo anlatımı (Özge Tektaş)
- 11:00 Egzersiz (Miro üzerinden paylaşımlar yapılacak)
- 12:00 Görüşme ve Tartışma

İkinci gün (26.11.2020) ATÖLYE V

- 09:30 Seminer (Nilüfer Kozikoğlu)
- 10:00 SpaceChase Game Anlatımı (Ömer Bilge Ersoy)
- 10:30 Egzersiz
- 11:30 Öğrenci Sunumları, Egzersize ilişkin Tartışma
- 12:00 Görüşme ve Tartışma

EK 20. Terimler Dizini

Mekan Dizimi Terimleri

Konfigürasyon: Belirli bir örüntü içinde karşılıklı bağlanmış tüm birimler arasındaki ilişkiler bütünü olarak tanımlanabilir.

Geçiş grafiği *(justified graph)*: Bir örüntü içindeki belirli bir noktadan diğer tüm noktalara olan derinliğe ilişkin bir resimdir (Hillier, 1996). Bu grafik, içinde bulunulduğu düşünülen mekanı başlangıç kabul ederek, mekanların daireler, aralarındaki ilişkilerin onları bağlayan doğrular olarak temsil edilmesi ve mekanların başlangıç mekanından olan derinliklerine göre bu mekanın üzerine yerleştirilmesi ile elde edilir.

Derinlik *(depth)*: Başlangıç olarak kabul edilen bir mekandan ulaşılmak istenen mekana giderken atılması gereken adım sayısıdır (Hillier ve Hanson, 1984).

Ortalama derinlik *(mean depth)*: Geçiş grafiğinde her derinlik adımında yer alan mekan sayısının toplamının tüm sistemdeki mekanların bir eksiğine bölünmesiyle elde edilen değerdir. Sisteme ait derinlik bilgisini verir.

Bağlılık *(connectivity)*: Her mekandan bir adım uzaklıkta olan mekan sayısını bildirir ve yan-yanalık, komşuluk ilişkisini tanımlar.

Entegrasyon *(integration)*: Bir mekanın entegrasyon değeri grafik içindeki tüm diğer mekanlardan o mekanın rölatif derinliğidir. Bir başka deyişle entegrasyon, o mekandan çizilen geçiş grafiğinin biçiminin sayısal bir ifadesini oluşturur (Hillier ve Hanson, 1984). Entegre mekanlar harekette zengin, ayrılmış mekanlar harekette fakir mekanlar olarak düşünülebilir.

Kontrol *(control*): Her mekanın ona doğru hareket eden komşusu için temsil ettiği seçim derecesini ifade eder. Mekanın kontrol değeri hesaplanırken her mekanın belirli sayıda, “n”, yakın komşusu olduğu ve her mekanın komşusuna 1/n değerini verdiği düşünülür. Belirli bir mekan için bu değerlerin toplamı o mekanın kontrol değerini verecektir (Hillier ve Hanson, 1984). Kontrol değeri yüksek olan mekanlar komşuları ile daha güçlü, kontrol değeri düşük olan mekanlar komşularıyla daha zayıf bağlantılara sahiptir.

Seçim *(choice/betweenness*): Bir mekanın diğer tüm mekanlar arasındaki en kısa yollarda çiğnenme sıklığıdır; bir mekan içinden akışı (flow) ölçer (Hillier ve diğ., 1987).

Girilebilirlik / Ulaşılabilirlik (*accessibility*): İki mekan arasındaki fiziksel hareket aracılığıyla ulaşılabilirlik ilişkisini tanımlamaktadır.

Görülebilirlik (*isovist*): Mekandaki belirli bir bakış noktasından belirli bir çevreye referansla görülebilen tüm noktaların bir kümesidir. Bir "isovist"in biçimi ve boyutu, pozisyona göre değişebilir (Benedikt, 1979).

Oyun Terimleri

Yönetici *(admin)*: Oyun kapsamında oyun tasarım ekibi ve araştırma projesi ekibi, yönetici/sistem yöneticisi olarak tanımlanmaktadır.

Yönetici ekranı *(admin paneli):* Sistem yöneticileri, sahne ve parametre yönetim arayüzüne erişebilmektedir. Bu arayüzü kullanarak sistem yöneticileri, uygulama içinde kullanılabilecek yeni sahneler yaratabilmekte, uygulama içinde kullanılan algoritmalar arasında seçim yapabilmekte ve parametrelerde hızlıca değişiklikler yapabilmektedirler.

Oyuncu: Mimarlık disiplini içinden veya dışından, oyun ekranına erişebilen, oyunun hedefleri doğrultusunda mekansal ilişkiler kurmayı deneyen herkes “oyuncu” olarak tanımlanabilmektedir.

Senaryo: Oyun tasarımı kurgusu gereği oyunu meydana getiren sahne ve bölümleri oluşturacak olası hikayelerdir. Bu senaryoların, özgün bir kurguda, dikkat çekici, eğlenceli ve kimi durumlarda oyunu meydana getiren sıralı sahnelerin devamı niteliğinde olması amaçlanmıştır. Oyun sahneleri ve bölümleri, senaryolardan yola çıkılarak oluşturulmaktadır.

Sahne: Oyun, farklı zorluk seviyelerine sahip bölümleri içeren sahnelerden oluşmaktadır.

Bölüm: Her bir sahne, kurgusu gereği farklı seviyelerde/hedeflerde bölümleri içermektedir.

Düğüm *(node)*: Oyunda eylemler ve bu eylemlere sahne olan mekanlar birer düğüm olarak temsil edilir. Düğümler birer mekan veya mekansal fonksiyonu gösterir.

Bağlamsal düğüm (*actuator / contextual node)*: Bir tür düğümdür ve düğümün tüm özelliklerini taşımaktadır. Bunların diğer düğüm noktalarından en büyük farkı ekranda sabit durmasıdır; özellikle bir fonksiyona işaret etmeleri gerekmez. Bağlamsal düğüm noktaları, bir mekanı temsil edebileceği gibi fiziksel bir sabit veya bir olgu da olabilir; örneğin, güneş, kuzey yönü, gürültü, otoyol, çöp kutusu birer bağlamsal düğüm noktası olarak tanımlanabilmektedir.

Bağ, bağlantı *(link/edge)*: Düğümler arası (düğüm-düğüm) veya bağlamsal düğüm-düğüm arası ilişkiyi sağlayan ara bağlardır. Bağlar oyuncu tarafından oluşturulur veya yok edilir.

Grasshopper Terimleri

Eklenti (*add-on, plug-in*): Grasshopper'ın standart kurulumunda bulunmayan bileşen paketleridir. food3rhino.com gibi web portalleri üzerinden indirilen bu paketler indirilerek Grasshopper içerisine yüklendiğinde sundukları özellikler ile Grasshopper'ın mevcut yetenekleri artmaktadır. Eklentiler kullanılarak oluşturulan Grasshopper kodlarını açabilmek için kodu çalıştıracak bilgisayarda da aynı eklentinin önceden yüklenmiş olması gerekir. Proje kapsamında geliştirilen SpaceChase for Grasshopper, bu nitelikte bir eklentidir.

Kangaroo: Grasshopper içerisinde geometrik optimizasyon yapabilen etkileşimli bir bileşen setidir. Geçmiş sürümlerde bir eklenti olan Kangaroo, Grasshopper'ın son sürümünde onun bir parçası haline gelmiştir.

Veri akış diyagramı (*dataflow diagram*): Grasshopper'ın dahil olduğu görsel programlama paradigmasının adıdır. Satırlar halinde çalıştırılan yazılı (text-based) kodlar yerine, ağ yapısına benzer bir düzende diyagramlar üzerinden gerçekleştirilen kodlama yöntemidir. Veri akışı kodlama paradigmasının (*data flow programming*) geleneksel kodlama paradigmasından (*control flow programming*) en büyük farkı, işlemlerin sıralamasının değil, verinin dönüşümünün tasarlanmasını temel almasıdır.

Kod (*definition*): Grasshopper'da oluşturulan kodlar, .gh uzantılı dosyalar biçiminde saklanır ve aktarılırlar. Bu kodlara "definition" adı verilmektedir. Bu çalışmada bu terim genellikle Grasshopper kodu olarak Türkçe'ye çevrilmiştir.

Bileşen *(component*): Grasshopper çeşitli işlevlere sahip kod parçalarının birbirleriyle ilişkilendirildiği veri akış diyagramlarına dayanır. Bu diyagramlarda kullanılan her bir kod parçasına bileşen (*component*) adı verilmektedir. Bir eklenti genellikle birden fazla bileşen gerektirir ve eklentinin mimarisi düşünülürken hangi bileşenlerin kodlanacağı ve birbirleriyle nasıl ilişkilendirilecekleri öngörülmelidir.

Girdi *(input*): Grasshopper'da bileşenlerin çalışması için veri ile beslenmeleri gerekir. Bileşenlere işlenmek üzere aktarılan verilere girdi *(input*) veya girdi parametreleri (*input parameters*) adı verilir. Ekran üzerindeki bileşen görselleştirmelerinde girdiler genellikle bileşenin sol tarafından bağlanırlar.

Çıktı *(output*): Grasshopper bileşenlerinin girdi verisini işlemesi sonucunda, başka bileşenlerde kullanılmak üzere sunduğu yeni verilere çıktı (*output*), veya çıktı parametreleri (*output parameters*) adı verilir. Ekran üzerindeki bileşen görselleştirmelerinde çıktılar genellikle bileşenin sağ tarafından alınarak başka bileşenlere bağlanırlar.

Kanvas (*canvas*): Grasshopper arayüzünde bileşenlerin yerleştirildiği boş alandır.

Hedef (*goal object*): Kangaroo geometrik optimizasyon bileşenlerinin girdi olarak kullandığı verilerdir. Örneğin, bir çizginin uzunluğunu belirli bir değere getirmek, bir noktanın koordinatının değişmemesini sağlamak veya bir yüzeyin düzlemsel kalmasını sağlamak gibi hedefler Kangaroo tarafından eş zamanlı olarak işlendiğinde, ortaya bu hedeflerin gerçekleşirken birbirleriyle çatıştığı durumlarda devingen bir animsyon çıkmaktadır.

Grup *(group*): Grasshopper'ın kullandığı veri akışı programlama paradigmasının dezavantajlarından birisi, bilgisayar ekranlarında iki boyutlu bir görsel dili temsil etmekte karşılaşılan sınırlardır. Ekranda aynı anda anlaşılır biçimde bulunabilecek bileşen sayısı sınırlıdır ve bu sınır "Deutsch limit" olarak adlandırılmaktadır. Grasshopper gibi programlama dilleri, bu sınırlılığı aşabilmek için çeşitli araçlar sunarlar. Bileşenlerin görsel olarak gruplanmaları Grasshopper kodlarının daha okunaklı olmasına yardımcı olan bir yöntemdir.

Küme (*cluster*): Gruptan daha ileri düzey bir kontrol ve düzenleme yöntemi olan kümeleme, programcının belirli bileşen paketlerini ekranda tek bir bileşenmiş gibi göstermesine olanak sağlamaktadır. Klasör mantığına yakın biçimde, kullanıcılar kümelere çift tıklayarak içlerindeki alt Grasshopper kodlarını açabilirler. Kümeler ayrıca, şifreleme gibi özellikler sunarak Grasshopper bileşenlerinin korunması işlevini de görürler. ShapceChase for Grasshopper'ın erken prototiplerinin sonuncusu (Prototip #4) bu nitelikte kümeler biçiminde organize edilmiştir.

Optimizasyon, eniyileme (*optimization*): Kangaroo bileşenlerinin, verilen hedefleri sağlaması geometrik bir optimizasyon (eniyileme)dir. Bu işlem eğer bir optimum sonuca ulaşırsa Kangaroo bileşeni *converged* (yakınsamış, belirli bir hedefe ulaşmış) mesajını verir ve optimizasyon işlemini durdurur. SpaceChase for Grasshopper, Kangaroo'nun optimizasyon hedefine ulaşmasını engellemek için çeşitli girdi parametrelerini kullanır ve böylelikle sistemin durmadan devinimi sürdürmesini sağlar. Bu çalışmada Türkçe'de yaygın kullanıma sahip olduğu için bu terim optimizasyon biçiminde kullanılmıştır.

Değişim günlüğü (*changelog*): Program geliştirme aşamalarında, sürümlerin (*version*) açıklandığı günlükler tutulur. Bu günlük dosyaları, yazılımda nelerin eklendiği, geliştirildiği, veya hangi hataların düzeltildiği bilgisini taşır. Böylelikle çok sayıda sürüm içerisinde hangi hataların ele alındığı veya hangi geliştirmelerin yapıldığı takip edilebilir ve numaralandırılan her yazılım sürümünün bir açıklama kaydı tutulmuş olur. SpaceChase for Grasshopper eklenti geliştirme sürecinde çok sayıda sürüm ortaya çıkmıştır. Bu sürümlerdeki farkları, gelişmeleri ve hata ayıklamalarını açıklayan değişim günlüğü, **EK X11**'de sunulmuştur. Bu sürümler içerisinde kritik öneme sahip olduğu için seçilen üç tanesi de bu raporun 4.1 numaralı bölümünde açıklanmıştır.

En kısa yol (*shortest path*): Hesaplamalı geometrinin (*computational geometry*) konu edindiği matematiksel bir sorundur. Bu proje bağlamında bir ağ yapısında herhangi bir düğüm

noktasından başka bir düğüm noktasına en az adımla giden yolun hesaplanmasıdır. Bu hesaplama pek çok Mekan Dizimi metriğinin temelini oluşturmaktadır. Bu çalışmada Dijkstra'nın geliştirdiği en kısa yol algoritması kullanılmıştır.

Metaball: Bilgisayar grafiğinde organik görünümlü n boyutlu yüzeylere verilen genel isimdir. Grasshopper'ın Metaball isimli bileşeni, verilen noktalara göre bir hesaplama yaparak iki boyutlu akışkan yüzeyler türetmektedir. Bu yüzeyler SpaceChase for Grasshopper'ın prototip aşamasında kullanılmıştır.

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Doç. Dr. PELİN DURSUN ÇEBİ
Proje No:	119M082
Proje Başlığı:	Mekan Dizimi Tasarım Uygulaması
Proje Türü:	1001 - Araştırma
Proje Süresi:	15
Araştırmacılar:	NİLÜFER KOZİKOĞLU, ABDURRAHMAN TUĞRUL YAZAR
Danışmanlar:	MUSTAFA DALCI
Projenin Yürütüldüğü Kuruluş ve Adresi:	İSTANBUL TEKNİK Ü. MİMARLIK F. MİMARLIK B.
Projenin Başlangıç ve Bitiş Tarihleri:	01/09/2019 - 01/12/2020
Onaylanan Bütçe:	426887.0
Harcanan Bütçe:	341659.46
Öz:	Mekan Dizimi, mimarlıkta ağ düşüncesine odaklanan, grafik-teorik araçlarla mekansal biçimlenmelerin ürettikleri potansiyelleri deşifre etmeye çalışan, mimarlık ve yapılı çevreye ilişkin bilimsel, veri odaklı, teorik bir kavrayış sunmayı hedefleyen bir teknikler bütünüdür. Mekan Dizimi araştırmalarının mimari tasarımda kullanılabilirliği ilgi çekici olmakla birlikte, bu tür bilimsel araçların tasarım sürecinde üretken bir biçimde kullanımına yönelik çalışmalar sınırlıdır. Araştırma projesi, bu boşluğu doldurmayı hedeflemekte, Mekan Diziminin gerçek zamanlı bilgi üreten, dinamik, yaratıcı bir araç olarak mimari tasarım sürecinde kullanılabilirliğini sorgulamaktadır. Bu bağlamda projenin ana ilgisi, Mekan Dizimi kuram ve uygulamalarından yola çıkan, topolojik-geometrik mekansal kurgular türetebilen, bilimsel temelli ve oyunsallaştırılabilir bir bilgisayar yazılımının geliştirilmesi üzerinedir. Bu proje yöntem olarak öncelikle kuramsal altyapıyı zenginleştirmek üzere mekansal ağları deşifre etmeyi hedefleyen Mekan Dizimi odaklı mevcut bilgisayar yazılımlarının olanak ve sınırlamalarını irdeler. Ardından yeni yazılım/lar için kodlama çalışmalarına, ilgili kriter/algoritma ve simülasyonların gerçekleştirilmesine odaklanır. Geliştirilen oyunsallaştırılmış yazılım/lar, mimarlık öğrencileriyle ve araştırma grubu içinden ve dışından uzman kişilerle gerçekleştirilen atölye çalışmalarıyla test edilmiş, potansiyelleri ve sınırları değerlendirilmiştir. Araştırma kapsamında iki yazılım geliştirilmiştir: SpaceChase for Grasshopper ve SpaceChase. SpaceChase for Grasshopper, Mekan Dizimi ile ilişkili metriklerin dinamik ağ yapısı üzerindeki etkisini araştırabilmek üzere geliştirilen, mimar ve mimarlık öğrencisi kullanıcılara yönelik olarak tasarlanan, Rhinoceros / Grasshopper yazılımı içerisinde çalışan bir eklentidir; oyunsal yazılıma bir ön hazırlık olarak düşünülebilir. Bu eklentiye paralel olarak tasarlanan, mimarlar dışında daha geniş bir kullanıcı kitlesine de ulaşmayı hedefleyen SpaceChase ise mekansal birimleri ve bağlama ait verileri düğümler aracılığıyla temsil eden, aralarındaki ilişkileri kurup bozarak olası bir araya gelmeleri, konfigürasyonları araştıran ve arka planda tüm olasılıkları Mekan Dizimi metrikleri ile değerlendirilebilir ve ölçülebilir hale getiren, bilim odaklı oyunsal bir yazılımdır. Akademiden ve uygulamadan çeşitli uzmanların ortaklığında gerçekleştirilen ve mekansal organizasyonların ölçülmesine, bir tasarım araştırması olarak ele alınmasına odaklanan deneysel çalışmanın mimarlık eğitimi ve pratiği için özgün bir değer ürettiği düşünülmektedir.
Anahtar Kelimeler:	Mekan Dizimi, Kanıt Temelli Tasarım, Mimarlıkta Ağ Düşüncesi, Mimari Tasarım
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Evet
Projeden Yapılan Yayınlar:	1- Dynamic Architectural Canvas: Designing a Relational Mapping Based Architectural Design Tool (Bildiri - Uluslararası Bildiri - Sözlü Sunum),

TÜBİTAK