OYUNLAR İÇİN GRAMER TABANLI PARAMETRİK 3B MODEL GELİŞTİRME

GRAMMAR-BASED PARAMETRIC 3D MODEL GENERATION FOR GAMES

MEHMET BATUHAN KARAARSLAN

PROF. DR. HAŞMET GÜRÇAY

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Grafiği Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

Eylül 2022

Mehmet Batuhan Karaarslan'ın hazırladığı **"OYUNLAR İÇİN GRAMER TABANLI PARAMETRİK 3B MODEL GELİŞTİRME"** adlı bu çalışma aşağıdaki jüri tarafından BİLGİSAYAR GRAFİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Dr. Güven ÇATAK
Başkan . . . . . .

Prof. Dr. Haşmet Gürçay
Danışman . . . . . .

Doç. Dr. Burkay GENÇ
Üye . . . . . .

Bu tez Hacettepe Üniversitesi Bilişim Enstitüsü tarafından YÜKSEK LİSANS TEZİ olarak onaylanmıştır.

Prof. Dr. Arif ALTUN
Bilişim Enstitüsü Müdürü

ETİK

Hacettepe Üniversitesi Bilişim Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

... / ... / .....

Mehmet Batuhan Karaarslan

YAYINLAMA FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan **"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"** kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H. Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- ☐ Enstitü yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.
- ☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren .... ay ertelenmiştir.
- ☐ Tezim ile ilgili gizlilik kararı verilmiştir.

... / ... / .....

Mehmet Batuhan Karaarslan

ÖZET

OYUNLAR İÇİN GRAMER TABANLI PARAMETRİK 3B MODEL GELİŞTİRME

Mehmet Batuhan Karaarslan

Yüksek Lisans, Bilgisayar Grafiği

Danışman: Prof. Dr. Haşmet Gürçay

Eylül 2021, 87 sayfa

Yordamsal üretim, istenen bir çıktının, manuel olarak bir insanın yapmasındansa, algoritmalar ve bilgisayar gücü kullanılarak oluşturulmasıdır. İstenen çıktı bir çok şey olabilir. Müzik, üç boyutlu veya iki boyutlu modeller, model dokuları bunlardan sadece birkaçıdır. Bu içeriklerin çalışma zamanında oluşturulabiliyor olması, oyun dünyasında da yordamsal üretimin kendine yer bulması anlamına gelmiştir. Özellikle oyun dosyalarında çok yer kaplayan modellerin oyun oynanırken oluşturulması, geliştiriciler kadar oyuncular için de büyük bir avantaj haline gelmiştir. Parametrik üretim özelleşmiş bir yordamsal üretim şekli olarak nitelendirilebilir. Parametrik üretimin farklılaştığı nokta, içerik üretilirken kullanılacak parametrelerin olduğu gibi kullanıcıdan alınıyor olmasıdır.

Bu çalışma kapsamında, şekil gramerleri kullanılarak üç boyutlu uzay gemisi modelleri oluşturulacaktır. Şekil gramerleri, parametrik üretim yapan algoritma için bir kural setidir. Bu kural seti kullanılarak ana hatlarıyla birbirine benzeyen ancak detaylarıyla farklılaşan modeller üretilebilir. Bu çalışmada, şekil gramerleri yine parametrik olarak oluşturulan ilkel şekillerin oluşturulması ve birbirleriyle bağlanmalarını sağlar. Bu çalışma kapsamında tanımlanan ilkel şekiller silindir, elipsoid, dikdörtgen prizma, üçgen, koni ve torustur. İlkel

şekillerin her birisi kendi algoritmalarıyla, şekil gramerinde verilen sınırlar doğrultusunda üretilir. Bu şekillerin birleştirilmesiyle çok daha karmaşık ve sofistike modeller üretilebilir.

Bu çalışma Unity Oyun Motoru üzerinde gerçeklenmiştir. Bu platformun seçilme nedeni, kullanımının kolay ve erişilebilir olmasının yanı sıra, oyun geliştiricilerinin arasında popüler olmasıdır. Kullanıcı model yaratmak istediğinde bir düğüm grafiği kullanarak istediği şekilleri birbirine bağlayarak oluşan farklı şekilleri görebilecektir. İstediği şekilleri .fbx formatlı bir üç boyutlu obje olarak çıktı alabilecektir.

Şekil gramerlerinin kullanıldığı birçok uygulamada, gramer kurallarının yazımının kodlama gibi bir arayüzle yapıldığı görülmüştür. Bu durum ise yordamsal üretim yapmak isteyen ancak kodlamayı göz korkutucu bulabilecek insanlar için de bir sorun teşkil etmektedir. Bu çalışma sonucunda çıkan yazılımda şekil gramerleri grafiksel ve daha kullanıcı dostu bir şekilde insanlara sunulacaktır.

Bu çalışma kapsamında geliştirilen yazılımın oluşturduğu modellerin çeşitlilikleri hem aynı şekil grameri için hem de farklı gramerler için incelenmiş ve yeterli olduğu yorumuna ulaşılmıştır. Aynı şekil grameri için şekiller yeterince benzer ancak ayrıntısal olarak farklı, farklı şekil gramerleri içinse tamamıyla farklılaşacak modeller oluştuğu görülmüştür. Bu şekil gramerleri sonucunda oluşan modeller kullanılarak bir oyun yazılmış, ve bu oyun katılımcılara oynatılmış ve sonrasında bir anket yapılmıştır. Anket sonuçlarına göre oluşan modeller yeterince çeşitli ve ilgi çekici olarak görülmüştür.

Anahtar Kelimeler: Yordamsal Üretim, Parametrik Üretim, 3B Gramer Kuralları, Şekil Gramerleri, Voksel Modelleme

ABSTRACT

GRAMMAR-BASED PARAMETRIC 3D MODEL GENERATION FOR GAMES

Mehmet Batuhan Karaarslan

Master of Science , Computer Graphics

Supervisor: Prof. Dr. Haşmet Gürçay

September 2022, 87 pages

Procedural generation is a process for creating an output with alghoritms and computing power instead of manual labour of a human. This output can be many different things. Music, two or three dimension models and model textures to name a few. Being able to create these contents in runtime means that procedural generation found its way into the gaming industry as well. Especially generating models, which can take up quite a size in the game files, during gameplay became a big advantage for developers as well as players. Parametric generation can be defined as a specialized procedural generation. The way parametric generation differs from procedural generation is that while using a parametric generation algorithm, the user has to provide the generation parameters to the algorithm.

Within the scope of this study, three dimensional space ship models will be generated using shape grammars. Shape grammars are rule sets for the parametric generation algorithm. Using this rule set, models, which have similar outlines but different details can be generated. In this study, shape grammars are used for generating primitive shapes and connecting them to each other. The primitive shapes are defined as cylinder, ellipsoid, rectangular prism, triangle, cone and torus. Each of these primitive shapes are generated within the specified

dimensions using their own algorithims. Much more complicated and sophisticated models can be generated by combining these shapes.

This study has been made on the Unity Game Engine. The reason for choosing this platform is that it is easy to use and accessible as well as popular with the game developers. When generating a model, the user will connect desired shapes using a node graph. Then they will be able to see and choose different models and export desired ones as a .fbx file.

It was seen that, in most applications of shape grammars, creating the grammar rules are being done with an interface which resembles coding. This may pose a problem for people who want to use procedural generation but find coding somewhat intimidating. The shape grammars are presented with a graphical interface and a more user friendly way in the developed software during this study.

The generated models were inspected for variety with the same and different shape grammars and determined to be adequate. For the same shape grammars, models with similar outlines but different details were produced. For different shape grammars, models were completely different. Using models resulting from these shape grammars, a game was developed. This game was played by participants and a questionnaire was held. According to the results of this questionnaire, the models were sufficiently various and interesting.

Keywords: Procedural Generation, Parametric Generation, 3D Grammar Rules, Shape Grammars, Voxel Modelling

TEŞEKKÜR

Bana hayatımın hiçbir anında desteklerini ve sevgilerini eksik bırakmayan anneme, babama ve kardeşim Ayşe'ye,

Bu tezin yazım sürecinde bilgilerini ve sabrını benden esirgemeyen Sayın Hocam Prof. Dr. Haşmet Gürçay'a,

Ve çalışmam gerektiğinde beni aksine ikna eden, çalışmak istemediğimde de beni zorlayan bütün sevdiklerime,

teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER

KISALTMALAR

CGI	:	**C**omputer **G**enerated **I**magery
FPS	:	**F**rames **P**er **S**econd
GB	:	**G**iga**B**yte
MB	:	**M**ega**B**yte
MRI	:	**M**agnetic **R**esonance **I**maging
RAM	:	**R**andom **A**ccess **M**emory

1. Giriş

İçinde 3 boyutlu modeller barındıran oyun, film gibi medyaların ölçeklerinin büyümesiyle, bu medyalar için detaylı ve çok sayıda model tasarlamak zorlu bir iş haline gelmiştir. Birbirinden farklı, özgün ancak tanınabilir ve karakteri olan modeller tasarlamak ciddi bir işgücüdür. Bilgisayar destekli yordamsal içerik üretimi ile bu problemin ölçeği fazlasıyla küçülmektedir.

Yordamsal içerik üretimi, bilgisayar kullanılarak belli kurallar çerçevesinde, algoritmaların yardımı ile istenilen içeriğin oluşturulması anlamına gelmektedir. Parametrik içerik üretimi de parametrelerini kullanıcıların verdiği bir tür yordamsal üretimdir. Harita ve model gibi nesnelerin oyunun bellekteki boyutlarını büyütmeden kullanılabilmelerine olanak verdiği ve her oynanışta kolaylıkla yeni içerik üretilmesine olanak verdiği için, özellikle oyun geliştiricileri için popüler bir olgudur. Her seferinde farklı içeriklerin oluşturulabilmesi hem oyunun yaşam döngüsünün uzamasını sağlar, hem de oyuncular açısından farklı deneyimler yaşamalarına vesile olur. Bir model yordamsal olarak oluşturulurken kullanılan en yaygın yöntemlerden birisi, önceden hazırlanmış olan parçalar verilen kurallara göre bir bütün oluşturacak şekilde birleştirilip, ortaya bir model çıkarılmasıdır. Ancak bu durumda da önceden ana modellerin oluşturulması gerekmektedir. Bu durum da oyun geliştiricilere bir iş yükü oluşması anlamına gelmektedir. Büyük oyun geliştirici şirketlerde bu durum çok büyük bir sıkıntı oluşturmasa da, özellikle küçük bağımsız oyun geliştirici takımlarında üç boyutlu model hazırlayabilecek bir kişi eksik bile olabilir. Yordamsal üretim bu noktada geliştiricilere büyük bir destek noktası olabilir. Zemin, harita gibi çok daha rastgele olabilecek modeller içinse genellikle gürültü fonksiyonları kullanılmaktadır.

Bu tezde, kuralları verilen herhangi bir 3 boyutlu modeli oluşturacak bir parametrik içerik üreteci geliştirilecektir. Bu üreteç, vokseller kullanarak her modeli verilen parametrelere göre baştan yaratacaktır. Önceden bahsedilen küçük parçaların bir kişi tarafından modellenmesine ihtiyaç duyulmayacaktır. Üreteç voxelleri birleştirerek modelin temel parçalarını oluşturacak, daha sonra belirtilen kurallara göre bu parçalar birleştirilerek, 3

boyutlu model oluşturulacaktır. Bu tez kapsamında yapılması planlanan modeller uzay gemisi modelleri olacaktır. Uzay gemisi modellerinin seçilmesinin sebebi, bu modellerin çok fazla varyasyonu olabilmesine rağmen, tanınabilirliklerinin yüksek olmasıdır.

1.1. Tez Çalışmasının Kapsamı

Bu tez kapsamında, oyun geliştirmede sıklıkla kullanılan yordamsal içerik üretimi için, parametrik olacak şekilde bir araç geliştirilmesi amaçlanmıştır Bu araç, geliştiricilerin elle bir modelleme yapmadan, yalnızca şekil gramerleri kullanarak sofistike modeller oluşturulabilmesini sağlamaktadır. Şekil gramerlerinin görsel bir arayüzle hazırlanması hedeflenmiştir. Bu aracın altyapısı ve kullanım yeri Unity Oyun Motorudur. Yapılmış akademik ve ticari ürünler incelendiğinde, bu konudaki araştırmaların eksik kaldığı görülmüştür. Crumley, buna benzer bir çalışma yapmıştır.[17] Onun çalışmasında, modelin oluşturulduğu alan bir ızgara gibi düşünülmüş, ve model önce vektörel olarak oluşturulup, daha sonra vokselli bir hale getirilmektedir. Bu çalışmada ise öncelikle daha basit olan ilkel şekiller üretilip daha sonra bunlar birleştirilerek büyük modeller oluşturulmaktadır. Böylece oluşan modelde daha büyük bir serbestlik oluşmaktadır.

Ayrıca bu modellerin ilgi çekici ve çeşitli olup olmadığının anlaşılması için basit bir oyun geliştirilmesi hedeflenmiştir. Bu oyun bir odak grubuna oynatılıp bir anket yapılması yoluyla düşüncelerinin öğrenilmesi amaçlanmıştır.

1.2. Özgün Değer

Başka çalışmalar sonucu ortaya çıkarılmış olan yordamsal üç boyutlu model üreticilerinin çoğu önceden hazırlanmış küçük parçaları birleştirme üzerine kuruludur. Voksel tabanlı yordamsal üreticiler ise genellikle arazi oluşturmada kullanılmaktadır. Bu çalışmada, modellerin yapıtaşı diyebileceğimiz ilkel şekiller algoritma tarafından parametrik olarak vokseller kullanılarak oluşturulacağı için geliştirilen yazılım, diğer yordamsal içerik üreticilerinden farklı olacaktır.

Şekil grameri kullanan çoğu yazılım, kodlamaya benzer bir arayüz ile gramer kurallarının oluşturulmasını sağlamaktadır. Geliştirililen yazılım ise, kullanıcılara grafiksel ve erişilebilir bir arayüz sunduğu için kullanması daha kolay olacak ve daha geniş bir kitleye hitap edebilecektir. En popüler oyun motorlarından birisi olan Unity Oyun Motoru üzerinde geliştirildiği için de projenin ulaşılabilirliği yüksek olacaktır. Bu sayede özellikle bağımsız oyun geliştiriciler için önemli bir araç olması beklenmektedir.

2. Literatür Özeti

Yordamsal üretici sistemleri için yapılmış birçok akademik ve ticari araştırmalar bulunmaktadır. Ancak bu araştırmanın konusu olan, sadece gramer kuralları kullanılarak yordamsal içerik üretimi için yapılmış araştırmalar kısıtlı görünmektedir. Bu bölümde yapılmış ve yayınlanmış akademik ve ticari araştırmalar incelenecektir. İncelenen araştırmalardan bazılarının kapsamı bu araştırmadan farklı olsa da, bu araştırmaların yordamsal üreticilerin algoritmalarının veya voksel grafiklerin oluşturulma mantıklarının anlaşılması için değerli birer kaynak oldukları görülmektedir.

Bu konuyla ilgili en kapsamlı araştırmanın, Voxel-Space Shape Grammars[17] isimli çalışmada yapıldığı görülmektedir. Bu çalışmada oluşturulacak şekiller için öncelikle vektörel bir yapı oluşturulduktan sonra şekil, vokseller kullanılarak üç boyutlu bir şekil oluşturulmaktadır. Kullanılan algoritmada beş adım bulunmaktadır. İlk adım şekil gramerinin yorumlanmasıdır. Bu adım neredeyse bütün yordamsal üreticilerde ortak olan bir adımdır. İkinci adımda yorumlanan gramer kullanılarak voksellerden oluşan bir şekil üretilmektedir. Üçüncü adım, voksellerin detaylandırılma adımıdır. Bu adımda yapılan işlemler yine gramer kuralları kullanılarak uygulanmaktadır. Dördüncü adım, yaratılan şeklin bir mesh olarak üretilmesini ele alır. Bu işlem marching cubes algoritması kullanılarak uygulanmaktadır. Beşinci ve son adımda ise üretilen mesh'e post-processing (rötuş) uygulanmaktadır. Bu işlem isteğe bağlı olup, meshin vokselli görüntüsünden kurtulmasını sağlar.

Çoğu yordamsal üretici, önceden modellenmiş şekilleri birleştirerek yeni bir şekil üretme yoluna gitmiştir. Önceden modelleme gerektirmeyen üreticilerin çoğunun ise arazi üreten yazılımlar olduğu görülmüştür. Bu çalışmanın en bilindik olanı, dünyanın en popüler oyunlarından birisi olan Minecraft[1] oyunudur. Bu oyun, vokseller kullanarak, oyuncuya her yeni oyunda tamamen eşsiz bir dünya yaratmaktadır. Oyun dünyası her seferinde yepyeni dağlar, köyler, mağaralar, ormanlar ve daha fazlasını yaratmaktadır. Bu oyunun

[1] Oyunun websitesi: www.minecraft.net

hem ticari hem de teknik başarısı, onu çoğu araştırmanın odak noktası yapmıştır. Procedural Generation of Voxel Worlds with Castles[18] adlı araştırmasında Dusterwald, yordamsal olarak yaratılmış bir haritada, gerçekçi yerleştirilmiş bir kale oluşturmaya çalışmaktadır. Bu çalışmasında, özellikle kalenin yerleştirileceği haritayı yaratırken Minecraft örneğinden çokça yararlanmıştır. Minecraft oyununun yordamsal üreticisi oyun dünyasını yaratırken tamamen bir gürültü fonksiyonu (Perlin Noise) kullandığı için tamamen rastgele bir arazi üretmektedir[19]. Üretilen modelin tamamen rastgele olması bu araştırmanın kapsamının dışında kalmakla beraber, bu oyun ve onun hakkında yapılan araştırmalar, yordamsal üreticilerin çalışma mantığının anlaşılması için önemli bir kaynak teşkil etmektedir.

Üç boyutlu modeller oluşturan yazılımlarda ise çoğunlukla, yapı taşı olarak nitelendirilebilecek, modelin en küçük parçaları elle modellenmektedir. Bu yapı taşı modeller daha sonra gramer kuralları kullanılarak birleştirilmektedir. Bu teknik Procedural Modeling of Buildings[20] ve Shape Grammars, Procedural Generation Techniques for Virtual Cities[21] araştırmaları bu konuyu irdelemiştir. Bu araştırmalarda modeller yaratılırken kurallar basamak basamak işletildiği için, çok basit kurallarla bile şekil karmaşıklaştıkça çok farklı modeller ortaya çıktığı görülmüştür. Bu yaklaşım çoğunlukla bina gibi tasarımlar için kullanılsa da, bu araştırmanın asıl çıktısı olacak olan uzay gemileri için de yapılabilir.

Görüldüğü üzere bu araştırmanın konusunu tamamen kapsayan çalışmalar çok kısıtlıdır. Ancak bahsedilen çalışmaların her biri bu tez sürecinde kullanılacak farklı adımları kapsadığı için, birlikte bu proje için önemli birer kaynak olacaklardır.

2.1. Yordamsal Üretim

Yordamsal üretim (procedural generation), herhangi bir içeriğin algoritmik olarak yaratılması anlamına gelmektedir[17]. Bu tip yazılımlar yardımıyla çok az kullanıcı girdisi ile içerik üretilebilmektedir. Kullanıcıların yazılıma sağladığı girdiler, en sonda oluşan içeriği doğrudan etkilemek yerine, algoritmayı yönlendirmek için sağlanır. Bu girdilere örnek olarak, tohumlar (seeds) ve gramer kuralları verilebilir. Tohumlar, yordamsal üreticiye

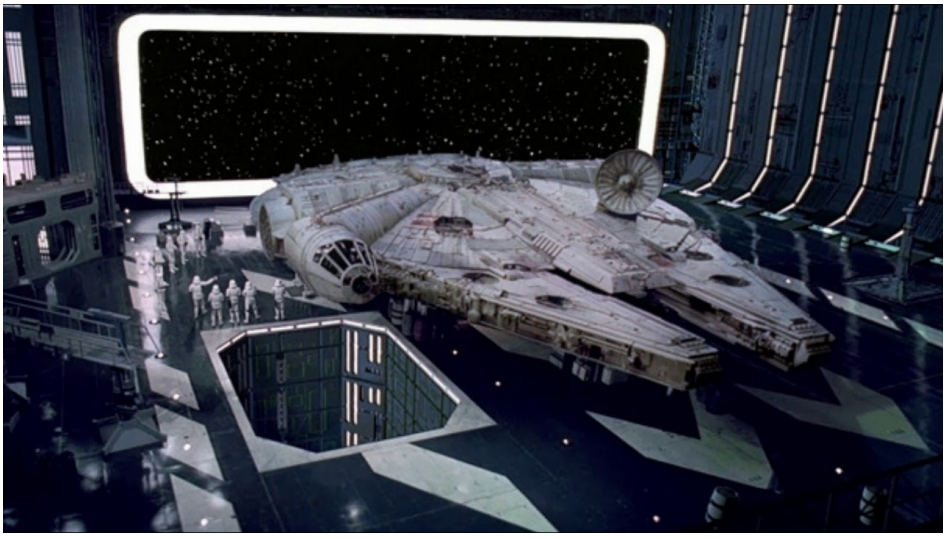
verilen bir koddur. Bu kod bir sayı veya harf dizisi olabilir. Algoritma bu girdiyi kullanarak farklı çıktılar üretebilir. Tohum kullanmanın avantajı, aynı tohum kullanıldığında her zaman aynı sonuç çıkmasıdır. Böylece üretilen bir içerik yalnızca tohum koduyla paylaşılabilinir veya tekrar ulaşılabilinir. Gramer kuralları ise, üreticiye verilen bir dizi komut anlamına gelmektedir. Algoritma bu kuralları kullanarak bir içerik üretir. Bu yaklaşımın avantajı ise, bir tek kural setinden birçok farklı çıktı oluşturulabilmesidir. Çok geniş veya çok sıkı kurallar verilerek çıktıların birbirlerinden çok farklı olup olmayacağı kontrol edilebilir. Bu yaklaşımın tohuma göre olan en önemli farkı, oluşan içeriğin her seferinde farklı olmasıdır. Bu farklılık duruma göre avantaj veya dezavantaj olabilir. Avantajı, üretilen çıktının varyasyonunun fazla olması, dezavantajı ise üretilen bir içerğin tohumu bir kere kaybedilirse bir daha ulaşımasının zor olabilmesidir.

Yordamsal üreteçlerin mutlaka rastgele içerik üretmesi gerekli değildir. Oyunda kullanılacak olan modellerin, oyun kurulum dosyası içinde boşuna yer kaplamasını engellemek için yordamsal üretimden fayda görülebilir. Modelin tam olarak nasıl yapılması gerektiğini anlatan dosyalar kullanılarak, algoritmik olarak üretilmesi de yordamsal içerik üretimi alanına girer. Buna örnek olarak mobil bir labirentten çıkma oyunu düşünülebilir. Geliştirici, oyunun içine konacak yüzlerce seviyeyi 3 boyutlu olarak modelleyip, özellikle mobil oyun dünyasında önemli olan hafızayı doldurmak yerine, bu modellerin bilgilerinin saklandığı bir veritabanı veya JSON dosyası kullanarak oyun çalışıyorken seviyeleri yaratabilir. Labirentin duvarlarının nereye geleceğinin koordinatlarını direkt saklandığı dosyadan alacağı için, her seferinde aynı şekil oluşturulabilir.

Yordamsal üretim kullanım amacı çoğunlukla hafızadan ve/veya zamandan tasarruf etmektir. Yaratılması gereken içerik önceden modellenmediği için programın çalıştırıldığı bilgisayarda hafıza işgal etmemektedir. Bu da, örneğin Minecraft gibi bir oyunun haritası pratikte sonsuzken bilgisayarda kapladığı yerin 1 gigabyte'ın altında olmasını sağlar. Bu durum, özellikle hafızanın çok değerli olduğu mobil oyun dünyasında çok önemlidir. Hafızayı daha verimli kullanan oyun ve uygulamalar, kullanıcıları için daha erişilebilir olmaktadır. Zamandan edilen tasarruf ise içeriği geliştirecek kişi açısından önemlidir. Bir medya geliştirme esnasında çok fazla veya çok büyük içeriklerin üretilmesi gerekiyorsa bunun elle

yapılmasındansa yordamsal olarak yapılması, sürecin hızlandırılmasını sağlayacaktır. Bu durum yordamsal üretimin sadece oyunlarda değil her türlü medyada kullanımının önünü açmaktadır.

Bahsedilen içerik geliştirme zorluklarına örnek olarak Star Wars filmleri verilebilir. 1977 yılında ilk defa yayınlanan Star Wars: A New Hope filminden beri bir çok özel efekte öncülük eden Star Wars serisinde araç ve yapı modelleri çok önemli yer kaplar. Bu modellerin her biri tasarımcı ve sanatçılar tarafından elle hazırlanmıştır[22]. Her biri ikonik olan bu modellerin tasarlanması ve ekrana taşınması çok büyük bir iş yüküdür. Tabi ki bu kadar ikonik olan araçların tamamıyla yordamsal olarak üretilebilmesi muhtemel değildir. Burada yordamsal üretime düşebilecek görev, modeli ana hatlarıyla oluşturduktan sonra tasarımcılara modeli geliştirebilmeleri için alan bırakmaktır. Örneğin, Şekil 2.1'de belki de film tarihinin en bilinen ve ikonik gemisi olan Millenium Falcon gösterilmiştir. Bu gibi bir geminin ana hatları yordamsal üretimle kolaylıkla oluşturulabilir. Daha sonra tasarımcılar, üstündeki küçük ayrınlıları kendi vizyonlarına göre değiştirip, modeli geliştirebilirler.



Şekil 2.1 Star Wars serisinden Millenium Falcon gemisi[1]

Bir başka kullanım alanı ise, çok fazla arka plan objesi olan sahneler için hızlı geliştirme imkanı sağlanmasıdır. Buna örnek olarak Serenity filmindeki son uzay savaşı sahnesi verilebilir. Bu gibi sahnelerde, arka planda çok fazla oluşturulması gereken model olabilir.

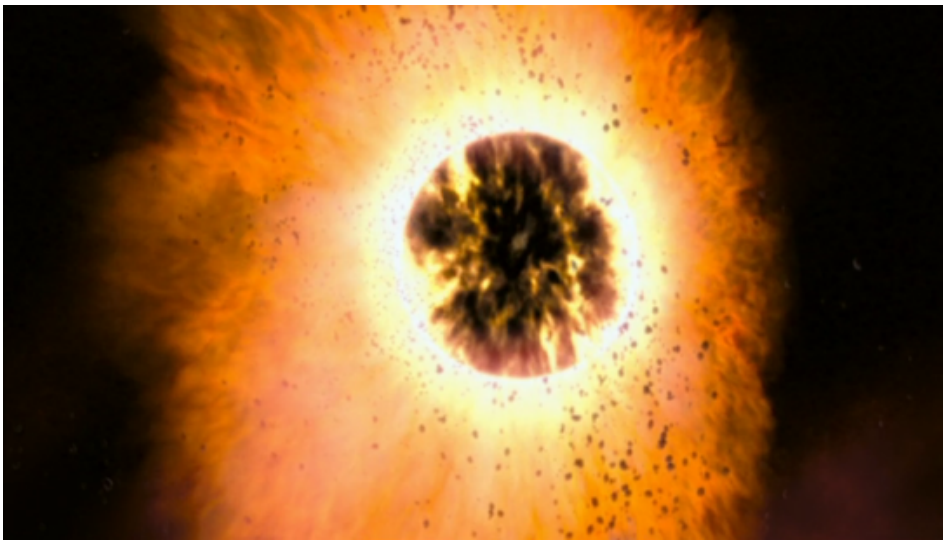
Tek gramer kuralı kullanılarak çeşitli gemiler yaratılabileceği için, bu modeller yordamsal olarak kolaylıkla oluşturulabilirler. Uzakta oldukları için çok fazla ayrıntıya da ihtiyaç duymadıkları için sonradan bir tasarımcı dokunuşuna da ihtiyaç duymayabilirler. Bundaki aşılması gereken sorun ise, gemilerin ”kültürel” farklılıklarıdır. Örneğin, Şekil 2.2'de bahsedilen Serentiy filmindeki uzay savaşından bir sahne verilmiştir. Burada karşıt tarafların gemilerindeki farklılık göze çarpmaktadır. Sağ taraftaki ”Alliance” gemileri, düzenli bir ordu oldukları için, askeri, nizami ve pragmatik olacak şekilde dizayn edilmişken, sol taraftaki ”Reaver” gemileri, vahşi korsanlar oldukları için, düzensiz, eski ve yıkık dökük olacak şekilde dizayn edilmişlerdir. Bu farklılıkları çözmenin yolu ise gramer kuralları kullanmaktan geçmektedir. Her iki taraf için de farklı ancak kendi içinde tutarlı gramer kuralları oluşturulursa, birbirinden farklı ancak bir bakışta hangi tarafa ait olduğu anlaşılacak modeller oluşturulabilir.

Şekil 2.2 Serenity filminden bir uzay savaşı sahnesi[2]

Şimdiye kadar, yordamsal üretimin gerçekte olmayan ve fantastik modelleri oluşturması üzerine bir anlatım yapıldı. Ancak bazen gerçek hayat, fanteziden çok daha ilginç ve gerçek dışı olabilmektedir. Buna bir örnek olarak Christopher Nolan'ın bir filmi olan *Yıldızlararası*'ndaki karadelik verilebilir[3]. Nolan, filminde neredeyse ayrı bir karakter olarak anılabilecek olan *Gargantua* isimli karadeliğin olabildiğince gerçekçi bir

şekilde beyaz perdeye taşınmasını istemiştir. Ancak karadeliklerin etraflarına uyguladıkları etkinin anlaşılmasının zorluğu sebebiyle, bunun klasik CGI teknikleri ile yapılamayacağını anlamış ve farklı bir yola gitmiştir. Karadeliğin modellenmesi için astrofizikçi Kip Thorne'dan yardım istemiş ve Thorne ve ekibi bu film için bir karadelik modelleme yazılımı hazırlamışlardır. Filmde görülen karadelik, tamamıyla bu yazılım tarafından yordamsal olarak oluşturulmuştur. Bu model o kadar gerçeğe yakın çıkmıştır ki, Thorne ve ekibi biri astrofizikçilere diğeri CGI artistlerine olmak üzere iki adet makale yayınlamıştır[23]. Klasik CGI yöntemleriyle yapılan karadeliklere bakıldığında, bu yöntemin ne kadar farklı sonuçlar oluşturduğu net bir şekilde görülebilir. Şekil 2.3 Yıldızlararası filmindeki karadeliği, Şekil 2.4 ise Doctor Who dizisindeki klasik CGI yöntemleriyle modellenmiş bir karadeliği göstermektedir.

Şekil 2.3 Yıldızlararası filmindeki Gargantua karadeliği[3]



Şekil 2.4 Doctor Who dizisindeki bir karadelik[4]

2.2. Parametrik Üretim

Parametrik üretim, yordamsal üretimin bir alt başlığı olarak nitelendirilebilir. İki içerik üretimi yönteminde de kullanıcıdan girdi alınması gerekmektedir. Ancak, yordamsal üretimde algoritmanın aldığı girdi kısıtlıdır. Örneğin, bir şekil grameri kural seti veya rastgele bir tohum dizisi böyle bir girdi olabilir. Bu durumda alınan girdi, algoritmanın çalışmasını dolaylı yoldan etkileyecektir. Çıkan içeriğe kullanıcının etkisi minimal düzeyde kalacaktır. Bu sayede çeşitlilik çok daha fazla artacaktır. Bu yaklaşım, arazi üretimi gibi durumlarda çok işe yarayacaktır ancak model üretiminde farklı zorluklara yol açabilmektedir. Parametrik üretimde ise, oluşan içerik, kullanıcının verdiği sınırlar ve parametreler arasındaki değerler ile oluşacaktır. Bu sayede kullanıcı oluşan içeriği daha doğrudan etkileyebilir.

Yordamsal model oluşturulurken, çoğu zaman modelin ayrıntılarının kontrol altında olması gereklidir. Yoksa şekil gramerleri kullanılmış olsa bile yalnızca rastgele parçalar birleşmiş, birleşen parçaların ise boyutları orantısız olabilir. Bu durumda kullanılabilir modellerden daha çok şekilsiz ve ilginç olmayan modeller oluşabilmektedir. Bu nedenle, model oluşturulurken kullanıcının daha çok kontrol sahibi olması gereklidir. Alışılagelmiş yordamsal üretimin kullanılması yerine parametrik üretim kullanılması ortaya çıkacak

şekillerin çeşitliliğini korurken, mantıklı ve istenen şekiller olmasına kapı açacaktır. Parametrik üretim, şekil gramerleriyle birlikte kullanıldığı zaman çok güçlü bir araç halini alabilir.

Bu tez kapsamında doğrudan yordamsal üretim yapan bir araç yerine parametrik üretim yapan bir araç geliştirilmesi, yordamsal üretimde oluşabilecek fazlaca rastgeleliği engellemek içindir. Parametrik üretim, oluşacak şekillere sınırlar koyulmasını gerektirdiği için daha kısıtlayıcı gibi görünse de, kullanıcıya vizyonunu daha net kullanmasını sağlayabildiği için çok daha özgürleştirici olduğu söylenebilir.

Şekil 2.5 Cyberpunk 2077 oyunundan yüz özelleştirme menüsü[5]

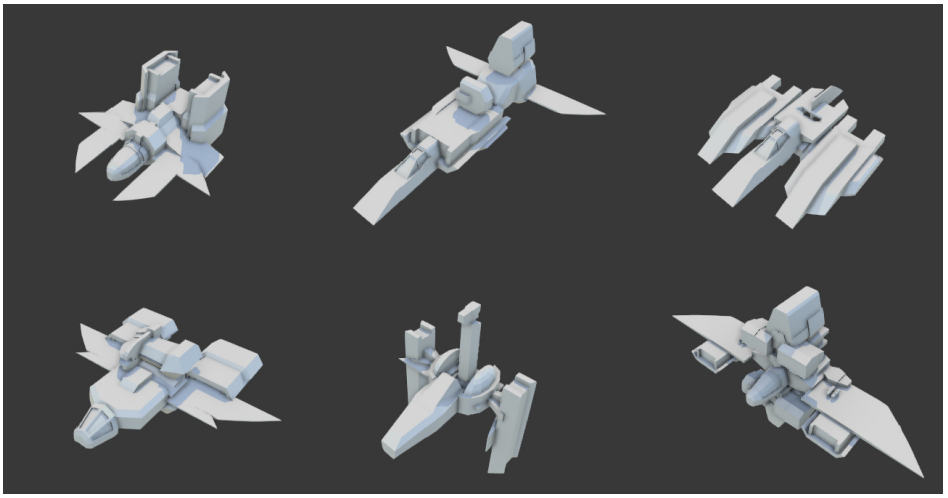
Şekil 2.5, Cyberpunk 2077 oyunundaki oyuncu karakterinin yüzünün özelleştirilebildiği menüyü göstermektedir. Oyuncular bu menü üzerinden istedikleri parametreleri değiştirerek, karakterin yüz hatlarının parametrik olarak oluşmasını sağlayabilmektedir. Karakterin yüzü verilen parametreler dahilinde yordamsal olarak oluşturulmaktadır. Bu sayede, yüzbinlerce farklı yüz modelleme gerektirmeden yaratılabilmektedir.

2.3. Şekil Gramerleri

Şekil gramerleri (Shape Grammars), yordamsal üretim algoritmalarına verilen bir dizi talimat olarak tanımlanabilir. Bu tezin konusu da daha önce bahsedilen tohumlu yordamsal üretim

yerine şekil gramerleri yardımıyla çalışan parametrik üreticilerdir.

Şekil gramerleri, basit birkaç şeklin, birbirleriyle olan ilişkilerini bir kural örüntüsü halinde temsil edilmesidir. Bu kurallar sayesinde temel şekiller kullanılarak çok daha karmaşık şekiller üretilebilmektedir[17].



Şekil 2.6 Önceden modellenmiş birkaç basit şekil ile üretilmiş 3 boyutlu modeller[6]

Şekil gramerleri üç adet bileşenden oluşur; temel şekillerin kümesi (sözlük), aksiyom ve üretim kuralları. Sözlük, algoritmanın kullanacağı en basit ve temel şekilleri içermektedir. Bu şekiller genellikle önceden elle modellenerek kullanılır. Bu tez kapsamında, bahsedilen basit şekiller de algoritma yardımıyla yaratılacaktır. Aksiyom, gramerin başlangıç noktası olarak belirtilen şekildir. Bu şekil sözlük kümesinin bir üyesi olmalıdır. Aksiyom ile başlayan gramer, üretim kurallarında belirtilen talimatlar doğrultusunda, sözlükteki farklı şekiller eklenerek karmaşıklaştırılır. Her yeni eklenen şekil için, o şekille ilgili olan kural işlenmeye devam eder. Bu işlem tekrarlanarak devam eder. Eğer her işlemde bir tane şekil üzerinde işlem yapılıyorsa gramerin seri olarak işlendiği, birden fazla şekil üzerinde işlem yapılıyorsa paralel olarak işlendiği söylenebilir.

Şekil gramerlerinin çok önemli faydalarından birisi, aynı kurallar kullanılarak, birbirinden farklı ancak benzer modeller çıkarılabilmesidir. Bu da izleyici ya da oyuncunun baktığı nesnenin ne olduğu ve kimin olduğu gibi sorulara bir bakışta cevap bulabilmesi için

önemlidir. Örneğin, bir film izlenirken, bir şehir sokağı gösterilirse, o şehrin Paris mi yoksa Ankara mı olduğu bir bakışta anlaşılabilir. Her iki şehirde de apartmanlar olmasına rağmen, apartmanların farklı ayrıntıları, aradaki kültürel farklılıkları ortaya koymakta ve bir bakışta tanınabilmelerini sağlamaktadır. Şekil gramerleri kullanılarak, bu özellikleri içinde barındıran farklı modeller yaratılabilir.

(a) Bir Ankara Apartmanı[24]

(b) Bir Paris Apartmanı[25]

Şekil 2.7 Ankara ve Paris Apartmanları

2.4. L-Sistemler

L-Sistemler de bir tür gramer yorumlama sistemidir[17]. L-sistemlerde, harflerden oluşan bir sembol dizisi kurallar dizisi olarak yorumlanır ve ona göre bir şekil oluşturulur. Axiom adı verilen bir başlangıç noktası ve birkaç basit kural ile, oluşturulan harf dizisi gittikçe karmaşıklaşacağı için, görülen şekil de gittikçe karmaşıklaşmaktadır. L-Sistemleri diğer yöntemlerden ayıran özelliği kuralları paralel olarak işlemesidir. Bu nedenle, özellikle, bitkilerin ve fraktalların modellenmesinde sıkça kullanılır. L-Sistemleri ilk geliştiren kişi olan Aristid Lindenmayer de bu sistemi bitki hücrelerinin davranışlarını ve bitkilerin gelişimini açıklamak ve araştırmak için kullanmıştır[26]. Doğadaki yosun ve bitki gibi canlılar da çok basit kurallar çerçevesinde çok kompleks şekiller ve davranışlar sergileyebildikleri için, L-Sistemler bu tür durumları sergilemek için idealdir. Örnek olarak, Lindermayer'in orijinal yosun büyüme modeli verilebilir;

Axiom: B

$A \rightarrow$AB

B$\rightarrow$A

Bu kurallara göre, oluşturulan her dizide, yeni dizi için, A görülen her yere AB, B görülen her yere ise A yazılması gerekir. Bu kurallar ve axiom kullanılarak, ilk sekiz adım oluşturulursa, B, A, AB, ABA, ABAAB, ABAABABA, ABAABABAABAAB ve ABAABABAABAABABAABABA kurallarının oluştuğu görülebilir. Lindenmayer'in araştırmasına göre bu modele göre büyümektedir. Eğer her bir adımın dizi uzunluğu da incelenirse, 1 1 2 3 5 8 13 21... şeklinde Fibonacci sıralamasının oluştuğu da görülebilir. Fibonacci sıralaması doğada ve fraktallarda çoğunlukla karşımıza çıkan bir olgudur.

2.5. Oyunlarda Yordamsal Üreticiler

Yordamsal üretim algoritmaları oyun geliştirme kapsamında genellikle arazi oluşturma için kullanılmaktadır. Bunun anlamı, oyuncular her yeni oyuna başladıklarında veya, hali hazırda oynadıkları oyunda haritayı keşfetmeye devam ettiklerinde, karşılarına her zaman başka bir içerik ve deneyimle karşılaşacak olmalarıdır. Bu durum geliştirilen oyunun hem kullanıcılar tarafından daha sıkça oynanmasını hem de oyunun yaşam döngüsünün daha uzun olmasını sağlar. Dünyanın en popüler oyunlarından olan Minecraft (Mojang) oyunu bunun en bilindik örneğidir. Minecraft oyununda yaratılan dünya blok blok yaratılmakta ve yordamsal olarak oluşturulmaktadır[19]. Yordamsal üretimin bu oyunda kullanımı, yaratılan oyun dünyasının pratikte sonsuz olmasını sağlamaktadır. Oyuncu her yeni oyuna başladığında yeni bir dünya yaratılmakta, oyunda toplanması gereken kaynakların yerleri ve miktarı değişmektedir. Böylece her dünya oyuncu için tamamen yeni bir başlangıç olmaktadır. Bu durum Minecraft 2011 yılında çıkmış olmasına rağmen hala tüm zamanların en çok satılmış oyunlardan birisi olmasını sağlamıştır[27].

Şekil 2.8 Minecraft oyunundan bir ekran görüntüsü[7]

Bir başka yordamsal üretimin önemli olduğu oyun No Man's Sky[2] (Hello Games) isimli hayatta kalma-keşif oyunudur. Bu oyunda kullanılan algoritma sayesinde 18 kentilyon ($18 * 10^{19}$) tane gezegen, üzerindeki hayvan ve bitki örtüleriyle birlikte eşsiz bir şekilde keşfedilmeyi beklemektedir[13]. Oyundaki her bir gezegen oyuncular keşfettikçe yordamsal olarak yaratılmaktadır. Gezegenlerde yaşayan hayvanlar ve bitkiler de daha sonraki başlıklarda anlatıldığı şekliyle yordamsal olarak yaratılır. Ancak No Man's Sky oyunu yordamsal üretim ile oyun geliştirmek isteyen geliştiricilere bir uyarı niteliği de taşımaktadır. No Man's Sky, yaşam döngüsüne o kadar büyük bir vizyonla başlamıştır ki, vadettiği özelliklerin çoğu eksik kalmıştır. Oyundaki gezegenlerde hayvanlar ve bitkiler olmasına rağmen cansız hissettirmekte, bazen bazı gezegenler tekrarlıyormuş gibi bir his vermekteydi. Oyun yorumcuları, oyunu "geniş fakat sığ" diyerek betimlemişlerdir[28]. Ancak Hello Games, oyunu unutulmaya terketmek yerine, üzerinde çalışmaya devam etmiş ve hatalarını düzeltmeye çalışmıştır. Oyun 2016'da yayınlanmasından beri birçok güncelleme almış ve baştaki vizyonuna uygun bir oyuna dönüşmüştür. Vadettiği özelliklere sonradan kavuşmuş olsa da, bu oyunun hikayesi oyun geliştiricilere bir örnek olmalı ve yordamsal üretimin direkt iyi içerik anlamına gelmediği anlaşılmalıdır.

[2]Oyunun websitesi: https://www.nomanssky.com/

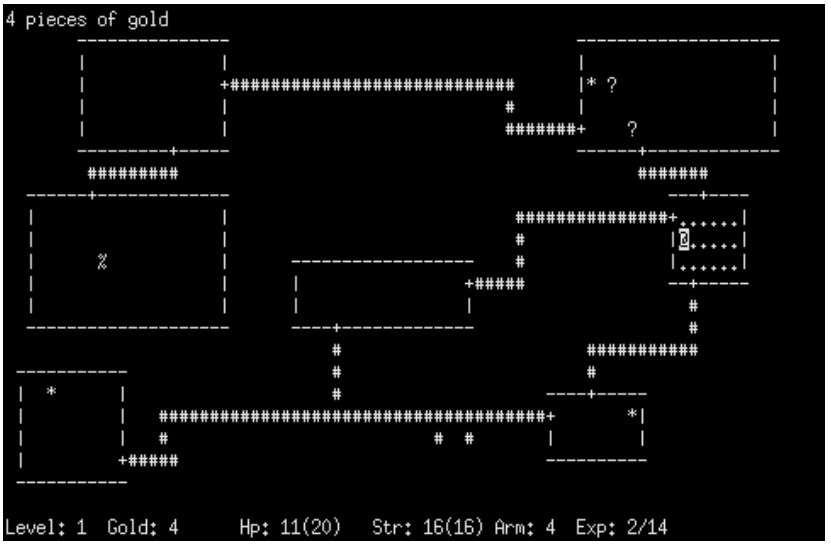
Şekil 2.9 No Man's Sky oyununda yordamsal olarak yaratılmış bir yaratık ve gezegen.

Bazı oyunlarınsa çekirdek özelliği içeriklerinin yordamsal olarak oluşturulmasıdır. Adını türünün ilk örneği olan Rogue oyunundan alan Roguelike türündeki oyunların en belirgin özelliği haritaların yordamsal olarak yaratılıyor olmasıdır[29]. Genellikle bu tür oyunlarda bir zindan veya mağara (dungeon) yordamsal olarak yaratılır ve oyuncu bu seviyelerden ilerlemeye çalışır. Eğer dungeon-crawler tarzı bir oyunun zindanları yordamsal olarak yaratılmamışsa bu oyun roguelike olarak kategorize edilemez. Bu tarz oyunlara örnek olarak Rogue (Michael Toy), NetHack (NetHack DevTeam), The Binding of Isaac (McMillen ve Himsl) ve Risk of Rain (Hopoo Games) verilebilir.

2.5.1. Yordamsal Harita Oluşturma

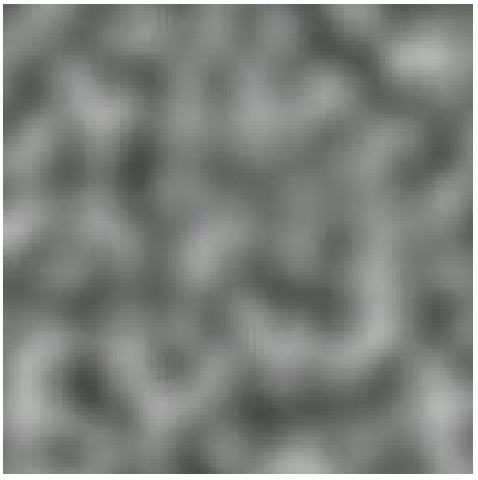
Oyunlarda kullanılan haritaların yordamsal olarak oluşturulması, yeni bir yöntem değildir. *Rogue*, 1980 yılında terminalden oynanan bir macera oyunu olarak çıkarılmıştır. Oyundaki amaç odalardaki canavarları yenip, hazineleri toplayıp seviye atlamak ve en son seviyedeki hazineye ulaşmakır. Ancak Rogue'un oyun tarihinde önemli olmasının sebebi yordamsal üretimi ilk uygulayan oyunlardan birisi olmasıdır. Oyuncunun oynadığı seviyelerin tamamı algoritmik olarak üretilmiştir. Bu oyundaki grafikler çok basit oldukları için algoritmanın

yapması gereken tek şey, seviyedeki odaları oluşturmak, canavarları ve hazineleri odalara yerleştirmek ve odaları koridorlarla birbirine bağlamaktır[29].



Şekil 2.10 Rogue oyununun ekran görüntüsü[8]

Bir başka yordamsal harita oluşturma yolu ise arazi oluşturmaktır. Özellikle açık-dünya oyunlarında, oynanan her oyunda farklı bir harita oluşması ve oyun deneyiminin farklılaşması için yordamsal üretim kullanılmaktadır. Arazilerin doğası gereği, arazi üzerindeki değişimler kademeli olmaktadır. Bu nedenle arazi üretiminde en çok kullanılan yöntem gürültü fonksiyonları kullanımıdır. Ancak bu gürültü tamamen rastgele olamaz. Bu sebeple, kademeli bir değişim oluşturan Perlin Gürültüsü kullanılmaktadır[17][19].



Şekil 2.11 İki boyutlu Perlin gürültüsü[9]

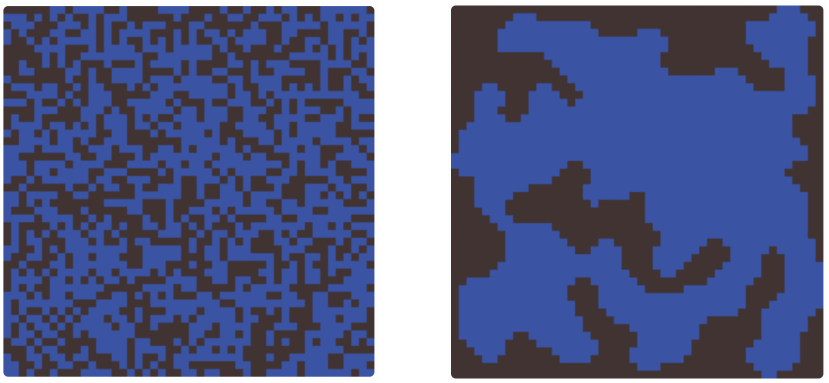
Perlin gürültüsü kademeli olarak değişen bir gürültü türüdür. Bu gürültü kullanılarak bir yükselti haritası oluşturulur. Gürültü grafiği üzerindeki her bir noktanın değeri, harita yaratılırken yükseklik değeri olarak alınır. Bu sayede arazilerde yer alan dağ, bayır, ova, deniz gibi doğal oluşumların oluşturulabilmesini sağlar. Bunun en bilindik örneği *Minecraft* oyunudur.

Şekil 2.12 Minecraft oyunundan bir ekran görüntüsü[10]

Perlin gürültüsü ile oluşturulan yükseklik haritalarının eksikliği detayların çok fazla görüntülenememesidir. Tek gürültü fonksiyonu kullanıldığında haritanın genel görüntüsü oluşmakta ancak taşlar, kayalar gibi detaylar oluşturulmak istenirse zorlanılmaktadır. Bunun çözümü ise farklı frekans ve genliklerde Perlin gürültüleri kullanılıp, üst üste bindirilmesidir. Böylece düşük frekanslı ve yüksek genlikli gürültü fonksiyonu haritanın genel hatlarını belirlerken, yüksek frekanslı ve düşük genlikli başka bir gürültü fonksiyonu ise haritanın ayrıntılarını belirler.

Bu tarz harita oluşturma yöntemi kullanılırken, haritanın yükselti bilgisi 2 boyutlu bir resimde tutulduğu için, yöntemdeki bir başka eksiklik daha göze çarpmaktadır. Mağara ve çıkıntı gibi oluşumlar bu yöntemle oluşturulamaz. Bu yapıları oluşturmak için farklı yöntemler kullanılmaktadır. Perlin solucanı ve hücresel otomasyon bu yöntemlerden ikisidir. Hücresel otomasyon kullanılarak oluşturulan mağaralar için öncelikle üç boyutlu bir ızgara üstünde rastgele vokseller oluşturulur[11]. Daha sonra her bir voksel için komşu vokseller

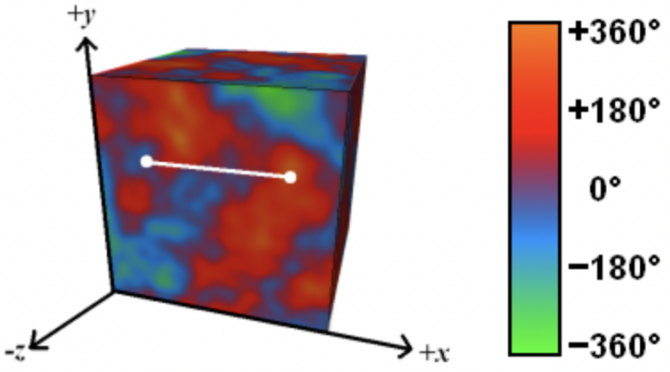
incelenerek, o vokselin haritada kalıp kalmaması gerektiği belirlenir. Örneğin bir voksel için 3 taneden az komşu voksele sahipse sil denebilir. Aynı işlemlerin yapıldığı birkaç yinelemeden sonra, istenen mağara modeline ulaşılır.



(a) Rastgele oluşturulmuş piksel ızgarası (b) 6 yineleme sonucu oluşmuş mağara

Şekil 2.13 Hücresel otomasyon kullanılarak 2 boyutlu oluşturulan bir mağara[11]

Perin solucanı yönteminde ise, harita oluşturmada kullanılan Perlin gürültüsü kullanılır[30]. Ancak kullanma yönteminde bir farklılık olmaktadır. Gürültü üzerindeki değişimler, yükselti değerlerini belirtmek yerine, solucanın gideceği yönü belirtmektedir. Örneğin, bir Perlin gürültüsü grafiği üzerindeki en yüksek nokta +180 derece dönüşü simgelerken, en düşük nokta -180 dereceyi simgeleyebilir. Buna göre oluşturulan bir mağara gerçekçi bir şekilde kıvrılarak uzayabilir.



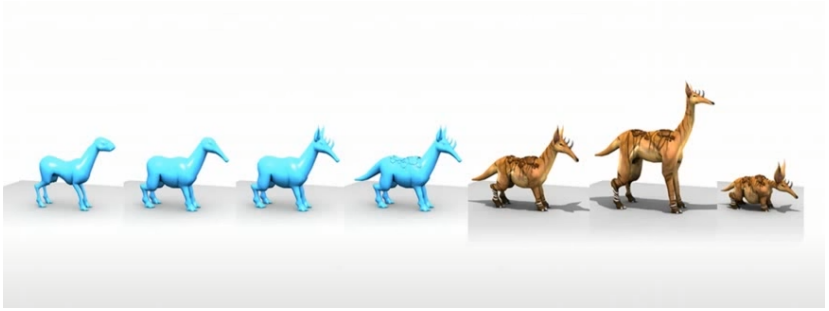
Şekil 2.14 Perlin solucanı oluşturmak için kullanılan bir gürültü[12]

Şekil 2.14'de bu işlem için kullanılabilecek bir Perlin gürültüsü görülmektedir. Gürültü değerlerinin kaç derece dönmeye karşılık geldiği şekildeki lejantta görülmektedir. Gürültü üzerindeki beyaz çizgi üzerinde ilerlenirken görülen değişime göre bir mağara oluşturulabilmektedir.

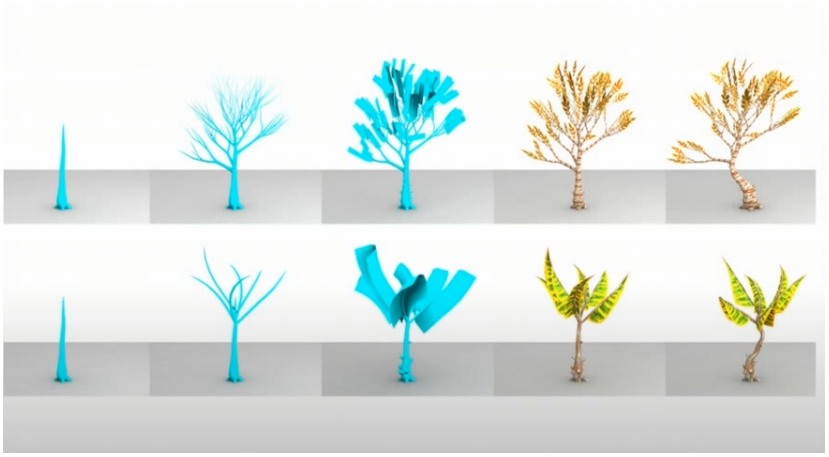
2.5.2. Yordamsal Model Oluşturma

Yordamsal model oluştururken, oluşturulacak modele göre aşılması gereken zorluklar da farklılaşmaktadır. Bir karakter modellenirken, karakterin uzuvlarının birbirinin içine geçmemesi, animasyonlarının gerçekçi durması, yakından da ayrıntılarının tatmin edici olması gibi gereklilikler varken, bir bina modellenirken, kapı pencere boyutlarının ve konumlarının mantıklı olması, içinde gezilebiliyorsa yeterince alan olması, oda düzeninin gerçekçi olması gibi farklı gereklilikler vardır. Bu nedenle yordamsal model oluşturmak için direkt kullanılması gereken algoritmalar bunlardır denilemez.

No Man's Sky isimli oyunda, yaratılan sayısız gezegenin yanı sıra, her bir gezegenin hayvan ve bitki örtüsünün de yordamsal olarak yaratılma gerekliliği ortaya çıkmıştır[13]. Bunu başarabilmek için, tasarımcıların izlediği yol da taslak sistemi dedikleri bir yöntemdir. Bu yöntemde, oyun tasarımcıları, canlılar için yüzlerce temel taslaklar hazırlamışlardır. Bu taslaklar, iskelet, duruş pozisyonu ve genel şekli kapsamaktadır. Ayrıca her bir taslak için etiketler de yapılmıştır. Gerçek hayatta iskelet sistemi birbirine benzeyen canlılar için aynı iskelet taslağı hazırlanmıştır. Eğer oyunda atabenzeyen ve ineğe benzeyen iki farklı canlı varsa aynı iskeleti kullanırlar. Aynı şekilde, yunus ve köpekbalığına benzeyen iki canlı da birbirleriyle aynı taslağı kullanacaklardır. Daha sonra bu canlıların yaşadığı habitata göre farklı aksesuarlar eklenir (pençe, boynuz yüzgeç gibi). Kemik kalınlığı ve boyut da rastgele değiştirildikten sonra canlının etiketine göre davranışları düzenlenir (örneğin kertenkele etiketli canlıların hepsi kertenkele davranışına sahip olur) ve canlı yaratılmış olur. Şekil 2.15 oyunda hayvanların oluşturulma adımlarını gösterirken, Şekil 2.16 oyunda bitkilerin oluşturulma adımlarını göstermektedir.



Şekil 2.15 No Man’s Sky oyununda hayvan yaratılması[13]



Şekil 2.16 No Man’s Sky oyununda bitki yaratılması[13]

2.6. Vokseller

Vokseller, piksellerin 3 boyutlu dünyadaki karşılığıdır. 2 boyutlu çalışmalarda pixeller kare olarak simgelenirken, 3 boyutlu çalışmalarda vokseller küptür. vokseller doğaları gereği ayrık (discrete) oldukları için, oluşturdukları şekilleri manipüle etmek çok daha kolaydır. voksellerin her biri üzerinde bir metadata (örn. renk, saydamlık, parlaklık gibi) tutabileceği için, oluşan şeklin görünüşünü algoritmik olarak değiştirmek de mümkündür[17].

Voksellerin bir diğer avantajı ise, farklı şekillerin birbirine eklenmesinin kolay olmasıdır. Mesh kullanılarak oluşturulan iki şeklin birbirine eklenmesi için, meshlerin birbirlerine dokundukları yerler bulunup, bu noktalarda yeni köşeler üretilip, oluşan şeklin yeni üçgenler kullanılarak çizilmesi gerekirken, voksellerden oluşan iki şekil için sadece yan yana koyulmaları yeterli olacaktır.

Voksellerin en çok kullanıldığı alanlardan birisi, tıbbi görüntülemedir. Vokseller, bir ızgara içindeki belirlenmiş üç boyutlu bir kesit alanın bilgilerini içinde tutabildiği için, bilgisayarlı tomografi, MRI, ultrason gibi işlemlerin çıktılarını oluşturmak için kullanılırlar[31].

Oyun sektöründe ise vokseller genellikle harita oluşturmada kullanılırlar. Bunun en bilindik örneği Minecraft oyunudur. Minecraft'ta bütün harita yordamsal olarak oluşturularak, vokseller ile gösterilir. Her bir voksel oyun içindeki bir kaynaktır ve kırılıp yerleştirilebilirler. Böylece vokseller aracılığyla oyuncular hayal edebildikleri her türlü yapıyı inşa edebilirler. Başka bir örnek ise Crysis oyunudur. Crysis, oyunculara vokselleri kullanma imkanı vermez ancak, oyunun kullandığı oyun motoru olan CryEngine 2 oyun geliştiricilerine bu aracı sağlar. CryEngine 2, çoğu oyun motoru gibi, arazi yükseklik bilgilerini bir yükselti haritasında tutmaktadır. Ancak, yükselti haritaları iki boyutlu resimler olduğu için, mağara ve çıkıntı gibi oluşumların bilgilerini tutamazlar. Bunun için Crytek, CryEngine 2'de mağara veya çıkıntı yapılmak istendiğinde bunların bilgilerini voksellerde tutmaktadır[32].

Şekil 2.17 Crysis oyunundan bir ekran görüntüsü[14]

3. Metot

Bu çalışmada, Unity Oyun Motoru üzerinde çalışan bir parametrik içerik üretici geliştirilmiştir. Üretilen içerik olarak, gramer kuralları kullanılarak özgün ancak tanınabilir uzay araçları yapılmıştır.

Oluşturulacak üç boyutlu modeller, ilkel şekillerin gramer kuralları çerçevesinde birleştirilmesi ile oluşturulmuştur. Bu ilkel şekiller, silindir, koni, dikdörtgen prizma, küre(elipsoid), torus ve üçgen olarak listelenebilir. Bu şekiller gramer kurallarında belirtilten aralıklar kullanılarak, algoritmik olarak oluşturulur. Şekiller oluşturulurken, öncelikle vokseller oluşturulur, verilen renk paletine göre her bir voksel renklendirilir ve birleştirilir. Daha sonra bu şekiller yine verilen gramer kuralları kullanılarak birleştirilir ve istenen şekil oluşturulur.

Tezin bu bölümünde, üç boyutlu modellerin oluşturulabilmesi için hangi adımların atıldığı anlatılacaktır.

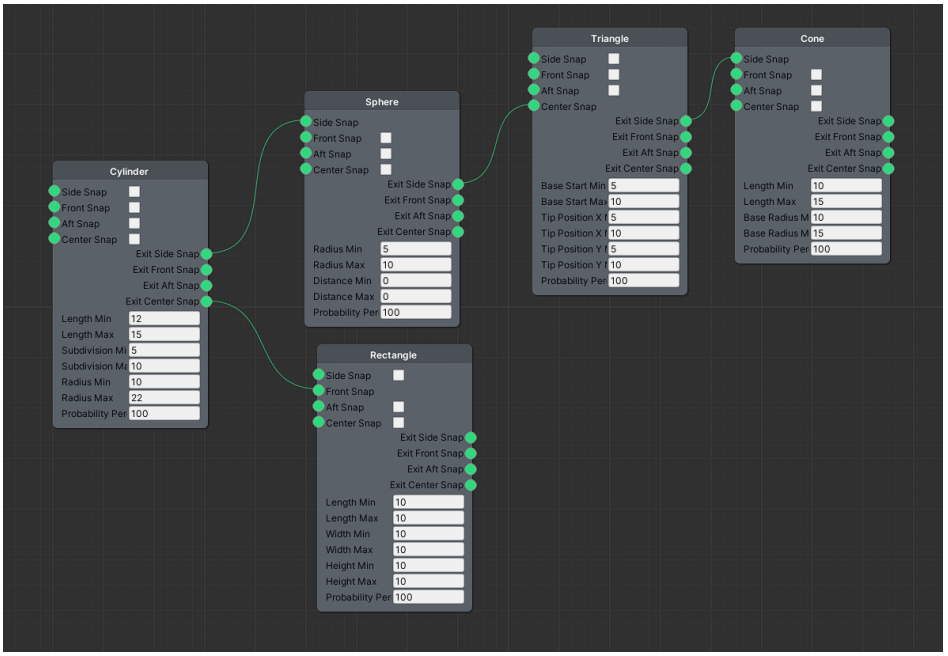
Takip edilen adımlar aşağıdaki gibidir;

1. Gramer kurallarının okunması,
2. Gramer kurallarına göre ilkel şekillerin oluşturulması,
3. Voksellerin oluşturulması,
4. Gramer kurallarına göre ilkel şekillerin birleştirilmesi.

3.1. Gramer Kurallarının Okunması

Gramer kuralları, parametrik üreticinin istenen objeyi yaratırken takip etmesi gereken kurallar bütünüdür. Bu kurallar, hem oluşturulacak ilkel şekillerin özelliklerini (boyutlarını), hem de nasıl birleştirileceklerini kapsar.

Gramer kuralları kullanıcıya kullanımı kolay ve görsel olacak şekilde bir düğüm grafiği şeklinde sunulmuştur. Düğüm grafiği, Unity Oyun Motoru için geliştirilmiş açık kaynak kodlu bir eklenti olan xNode kütüphanesi tarafından sağlanmıştır. Şekil 3.1, ilgili düğüm grafiği için bir örnek teşkil etmektedir.



Şekil 3.1 Örnek Gramer Kuralları Düğüm Grafiği

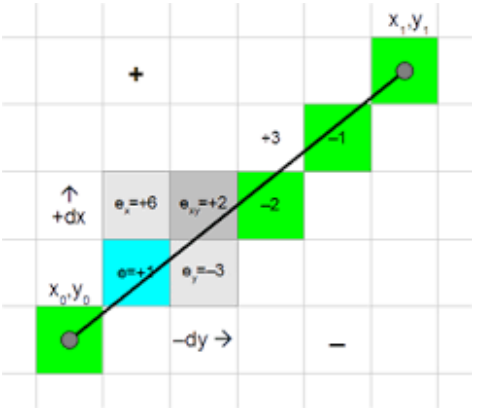
Her bir düğümde, ilgili şekle özel boyut bilgileri ve o şeklin oluşturulma ihtimali kullanıcı tarafından girilir. Boyut bilgileri, her bir bilgi için maksimum ve minimum olarak girilir. Bu, ortaya çıkan şekillere istenildiği ölçüde rastgelelik katar. Gramer kuralları okunduktan sonra, oluşturulacak şekiller bir listeye konur ve tek tek oluşturulur.

3.2. Gramer Kurallarına Göre Basit Şekillerin Oluşturulması

Bir şekil oluşturulacağı zaman, kendisine ait sınıfı tetiklenir ve şekli oluşturacak olan voksellerin koordinatları belirlenir. Bu işlemin algoritması her bir şekle özeldir. Voksel pozisyonları belirlenirken, şekillerin birbiri ile birleştirileceği bağlantı noktaları da belirlenir. Bu noktalar şeklin en üst ve en altı (ön ve arka bağlantı noktaları) ve yan kenarların en ortası (yan bağlantı noktaları) olacak şekilde seçilir.

3.2.1. Bresenham Algoritması

Bu çalışmada şekillerin oluşturulma sırasında Bresenham Algoritması çokça kullanılmaktadır. Bu nedenle şekillerin nasıl oluşturulduğu incelenmeden önce Bresenham Algoritması'nın ne olduğu ve nasıl çalıştığının anlatılması gereklidir.

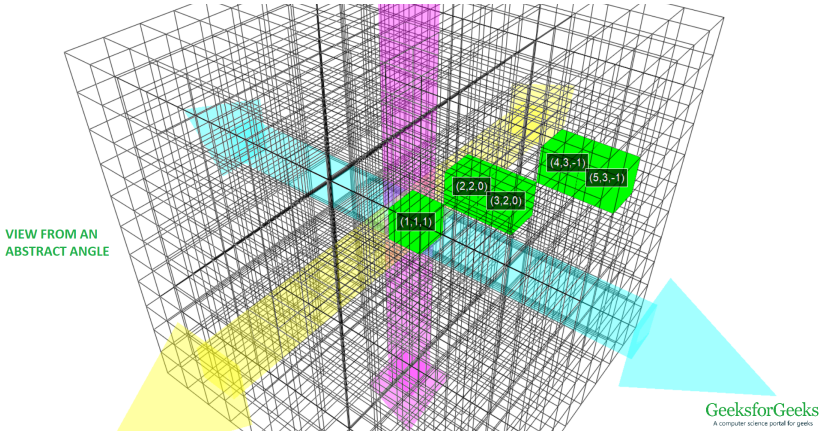


Şekil 3.2 Bresenham Algoritması ile 2 boyutlu çizilmiş bir doğru[15]

Bresenham Algoritması, iki nokta arasına çizilebilecek bir doğrunun yaklaşık halini pikselli olarak çizdirilebilmesini sağlar. Algoritma çok basit ve hızlı olduğu için bilgisayar grafiği alanında kullanımı sıklıkla görülebilir. Öyle ki, modern grafik kartlarının içinde bulunan donanımlara gömülü olarak bile kullanılmaktadır. Çok hızlı olmasının sebebi yalnızca toplama, çıkarma ve çarpma işlemlerini kullanıyor olmasıdır. Algoritmanın orijinali bilgisayar ekranları için iki boyutlu olacak şekilde geliştirilmiş olsa da, algoritmanın basitliği, üç boyuta göre değiştirilmesini oldukça kolaylaştırmaktadır. Brensenham Algoritması, elips, küre ve bezier eğrileri gibi şekillere de uyarlanabilir.

Algoritma şu şekilde çalışmaktadır; Öncelikle verilen iki nokta arasındaki gerçek doğru çizilir. Bu doğrunun eğimi kullanılarak iki nokta arasındaki pikseller için, gerçek doğruya en yakın pikseller seçilir. Kontrol edilecek bir sonraki piksel doğrunun ilerlediği yöne göre seçilir. Bu işlem, ilk noktadan başlanıp son noktaya ulaşıldığında biter. Böylece ekranda gösterilebilecek yaklaşık pikselli doğru kolaylıkla bulunabilir. Algoritma üç boyutlu ortamda kullanılacaksa, aynı işlemler üç boyuta uyarlanarak vokselli doğrular oluşturulabilir. Bu

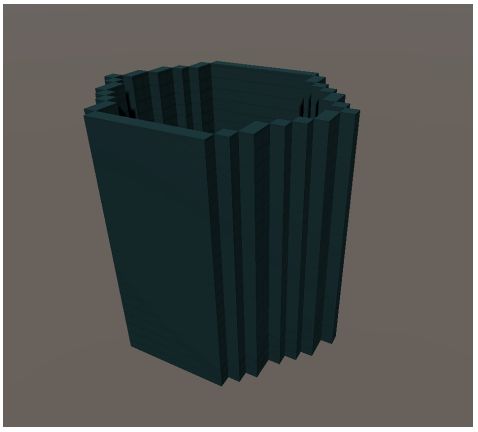
uyarlamada farklı olan işlem dominant olan ekseni bulmaktır. Bu ekseni bulmak için iki nokta arasındaki eğimin farklı eksenlerdeki değerleri karşılaştırılır. Örneğin x eksenindeki pozitif eğim en fazlaysa, bir sonraki voksel, (x+1, y, z), (x+1, y+1, z), (x+1, y, z+1) veya (x+1, y+1, z+1) olabilir. Böylece doğrunun üstünde bulunması imkansız olan vokseller kontrol edilmez. Bu çalışma kapsamında Brensenham Algoritması üç boyutlu olarak sıkça kullanılmıştır.



Şekil 3.3 Bresenham Algoritması ile 3 boyutlu çizilmiş bir doğru[16]

Bresenham Algoritması'nın basit ve hızlı olmasının bedeli, oluşan doğruların, algoritmadan anti-aliasing'li olmamasıdır. Oluşturulan doğru, ayrı bir anti-aliasing algoritmasına sokulursa ise zaman ve kaynak kaybı yaşanmaktadır. Eğer anti-aliasing gerektiren bir çözüm gerekiyorsa, Bresenham Algoritması'na benzer olan Wu Algoritması kullanılmalıdır. Wu Algoritması, Bresenham Algoritması'na çok benzemektedir. Aralarındaki fark, iki nokta arasındaki çizilen gerçek doğruya yakın olan pikseller, yakınlıklarına göre renklendirilir. Böylece oluşan doğru üzerinde anti-aliasing efekti de oluşmuş olur.

3.2.2. Silindir Oluşturulması



Şekil 3.4 Örnek Silindir

Silindir oluşturulabilmesi için düğüm grafiğinde kullanıcıdan istenen girişler aşağıdaki gibidir;

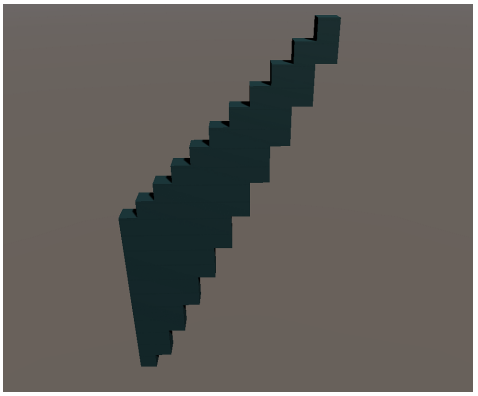
1. Maksimum uzunluk,
2. Minimum uzunluk,
3. Maksimum altbölüm sayısı,
4. Minimum altbölüm sayısı,
5. Maksimum yarıçap,
6. Minimum yarıçap.

Minimum ve maksimum değerler arasında birer değer seçilerek ilgili boyut değerleri belirlenir. Silindir oluşturulurken öncelikle, 360 dereceyi altbölüm kadar bölen ve merkeze yarıçap kadar uzak olan noktalar belirlenir. Daha sonra bu noktalar üç boyutlu Bresenham Algoritması kullanılarak birleştirilir ve bir çember oluşturulur. Bu çember uzunluk kadar uzatılır ve silindir elde edilir. Şekil 3.4, 20 voksel uzunluğunda, 10 voksel yarıçapa ve 6 altbölüme sahip bir silindiri göstermektedir.

Algorithm 1 Silindir

0: *length* ← length of *cylinder*
0: *sides* ← number of *subdivisions*
0: *radius* ← radius of *cylinder*
0: Set *length* with a random value between lengthMin, lengthMax
0: Set *side* with a random value between sidesMin, sidesMax
0: Set *radius* with a random value between radiusMin, radiusMax
0: **for** $side = 0, 1 \ldots$.of sides **do**
0: Divide 360 to sides count to find each slice angle.
0: Multiply *side* with resulting angle.
0: Calculate corner position with side angle and *radius*.
0: Add calculated positions to the positions list.
0: **end for**
0: **for** $corner = 0, 1 \ldots$.of corner positions **do**
0: Use Bresenham3D algorithm between the corners to connect them.
0: Add calculated positions to the positions list.
0: **for** each points in output of Bresenham3D algorithm **do**
0: Lenghten each point up to *length*.
0: Add calculated positions to the positions list.
0: **end for**
0: **end for**
=0

3.2.3. Üçgen Oluşturulması



Şekil 3.5 Örnek Üçgen

Üçgen oluşturulabilmesi için düğüm grafiğinde kullanıcıdan istenen girişler aşağıdaki gibidir;

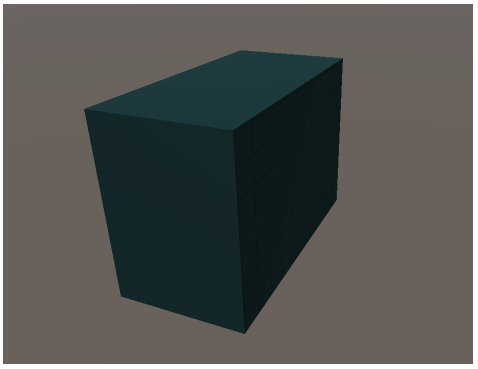
1. Maksimum taban başlama noktası,
2. Minimum taban başlama noktası,
3. Maksimum köşe X pozisyonu,
4. Minimum köşe X pozisyonu,
5. Maksimum köşe Y pozisyonu,
6. Minimum köşe Y pozisyonu.

Minimum ve maksimum değerler arasında birer değer seçilerek ilgili boyut değerleri belirlenir. Üçgen oluşturulurken öncelikle, üçgenin dik kenarı boyunca bir çizgi çekilir. Daha sonra dik kenarın en altı ve en üstünden köşe pozisyonuna birer çizgi çekilir. Bu çizgiler üç boyutlu Bresenham Algoritması kullanılarak çizilmektedir. En sonunda ise üçgenin içinde kalan alan da doldurularak üçgen çizilmiş olur. Şekil 3.5, taban başlama noktası 10, köşe X pozisyonu 20, ve köşe Y pozisyonu 10 olan bir üçgeni göstermektedir.

Algorithm 2 Üçgen

0: *baseStart* ← start point of *triangle*'s straight edge on y-axis.
0: *baseEnd* ← end point of *triangle*'s straight edge on y-axis.
0: *tipYValue* ← number of *subdivisions.*
0: *tipXValue* ← radius of *cylinder.*
0: Set *baseStart* with a random value between baseStartMin, baseStartMax.
0: Set *tipYValue* with a random value between tipYValueMin, tipYValueMax.
0: Set *tipXValue* with a random value between tipXValuesMin, tipXValueMax.
0: Set *baseEnd* to 0.
0: **for** $side = 0, 1 \ldots$.of sides **do**
0: Divide 360 to sides count to find each slice angle.
0: Multiply *side* with resulting angle.
0: Calculate corner position with side angle and *radius*.
0: Add calculated positions to the positions list.
0: **end for**
0: **for** $corner = 0, 1 \ldots$.of corner positions **do**
0: Use Bresenham3D algorithm between the corners to connect them.
0: Add calculated positions to the positions list.
0: **for** each points in output of Bresenham3D algorithm **do**
0: Lenghten each point up to *length*.
0: Add calculated positions to the positions list.
0: **end for**
0: **end for**
=0

3.2.4. Dikdörtgen Prizma Oluşturulması



Şekil 3.6 Örnek Dikdörtgenler Prizması

Dikdörtgen prizma oluşturulabilmesi için düğüm grafiğinde kullanıcıdan istenen girişler aşağıdaki gibidir;

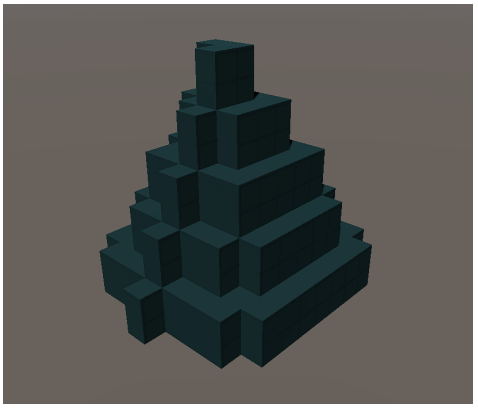
1. Maksimum uzunluk,
2. Minimum uzunluk,
3. Maksimum genişlik,
4. Minimum genişlik,
5. Maksimum yükseklik,
6. Minimum yükseklik.

Minimum ve maksimum değerler arasında birer değer seçilerek ilgili boyut değerleri belirlenir. Dikdörtgen prizma oluşturulurken genişlik ve yükseklik değerleri ile bir dikdörtgen çizilir. Daha sonra bu dikdörtgen uzunluk kadar uzatılır. Şekil 3.6, uzunluğu 10, genişliği 5 ve yüksekliği 7 olan bir dikdörtgenler prizmasını göstermektedir.

Algorithm 3 Dikdörtgen Prizma

0: *length* ← length of *Rectangle*'s straight edge on z-axis.
0: *width* ← width of *Rectangle*'s straight edge on x-axis.
0: *height* ← height of *Rectangle*'s straight edge on y-axis.
0: Set *length* with a random value between lengthMin, lengthMax.
0: Set *width* with a random value between widthMin, widthMax.
0: Set *height* with a random value between heightMin, heightMax.
0: **for** $x = 0, 1 \ldots width$ **do**
0: **for** $y = 0, 1 \ldots height$ **do**
0: **for** $z = 0, 1 \ldots length$ **do**
0: Add x,y,z point to the positions list.
0: **end for**
0: **end for**
0: **end for**=0

3.2.5. Koni Oluşturulması



Şekil 3.7 Örnek Koni

Koni oluşturulabilmesi için düğüm grafiğinde kullanıcıdan istenen girişler aşağıdaki gibidir;

1. Maksimum uzunluk,
2. Minimum uzunluk,
3. Maksimum taban yarıçapı,
4. Minimum taban yarıçapı.

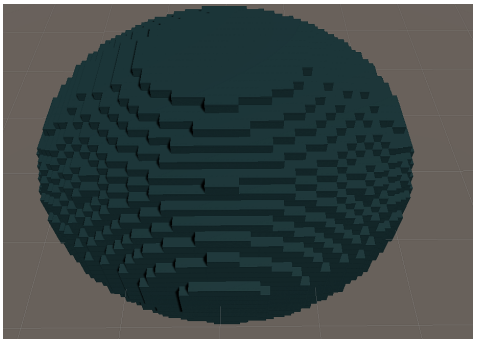
Minimum ve maksimum değerler arasında birer değer seçilerek ilgili boyut değerleri belirlenir. Koni oluşturulurken ilk olarak uç noktasından tabanda yer alacak bir noktaya

üç boyutlu Bresenham Algoritması kullanılarak bir çizgi çekilir. Daha sonra her bir uzunluk birimi için merkezden geçen eksene eşit uzaklıktaki noktalar alınır. Şekil 3.7, uzunluğu 10, altbölüm sayısı 8 ve taban yarıçapı 5 olan bir koniyi göstermektedir.

Algorithm 4 Koni

0: *length* ← length of *Cone*
0: *baseRadius* ← radius of *Cone*'s base circle.
0: Set *length* with a random value between lengthMin, lengthMax.
0: Set *baseRadius* with a random value between baseRadiusMin, baseRadiusMax.
0: Find the tip point of the cone.
0: Find a point on the base circle of the cone.
0: Use Bresenham3D algorithm between the points to connect them.
0: **for** each points in output of Bresenham3D algorithm **do**
0: Add points with equal distance to the centerline to the positions list.
0: **end for**=0

3.2.6. Elipsoid Oluşturulması



Şekil 3.8 Örnek Elipsoid

Elipsoid oluşturulabilmesi için düğüm grafiğinde kullanıcıdan istenen girişler aşağıdaki gibidir;

1. Maksimum yarıçap,
2. Minimum yarıçap,
3. Maksimum merkezler arası uzaklık,
4. Minimum merkezler arası uzaklık.

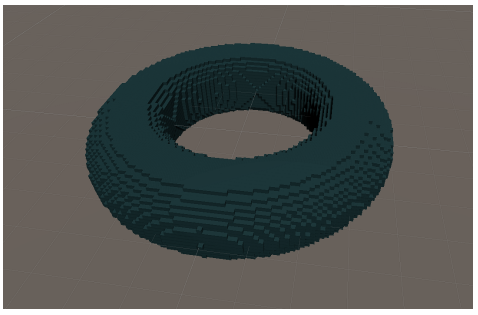
Minimum ve maksimum değerler arasında birer değer seçilerek ilgili boyut değerleri belirlenir. Elipsoid oluşturuluken ilk olarak yarıçap ve merkezler arası uzaklık kullanılarak semi-major ve semi-minor eksen hesaplanır. Eğer semi-major ve semi-minor eksenler birbirine eşit değilse, şekil üç boyutlu bir elips olmaktadır. Eğer oluşturulacak şeklin merkezler arası uzaklık değeri sıfırsa, semi-major ve semi-minor eksenler birbirine eşit ve yarıçap kadar olmaktadır. Bu durumda oluşan şekil küre olmaktadır.

Elips eksenleri hesaplandıktan sonra, iki merkeze de uzaklığı toplamı 2 semi-major eksenin iki katı olan noktalar bulunur. Şekil 3.8, yarıçapı 10 ve merkezler arası uzaklığı 25 olan bir elipsoidi göstermektedir.

Algorithm 5 Elipsoid

0: *radius* ← radius of *Ellipsoid*
0: *centerDistance* ← distance between two centers of ellipsoid
0: Calculate semi-major axis.
0: Calculate semi-minor axis.
0: **for** each points in a box which has dimensions as semi-major axis and semi-minor axis **do**
0: Find points with sum of distances to each center is 2 semi-majors.
0: Add points to the positions list.
0: **end for**=0

3.2.7. Torus Oluşturulması



Şekil 3.9 Örnek Torus

Torus oluşturulabilmesi için düğüm grafiğinde kullanıcıdan istenen girişler aşağıdaki gibidir;

1. Maksimum dış yarıçap,

2. Minimum dış yarıçap,
3. Maksimum iç yarıçap,
4. Minimum iç yarıçap.

Minimum ve maksimum değerler arasında birer değer seçilerek ilgili boyut değerleri belirlenir. Bu değerlerde dış yarıçap torusun çembersel kısmının yarıçapı, iç yarıçap ise torusun içte kalan boş kısmın yarıçapıdır. Şeklin içinde kalabilecek bütün noktalar için, çembersel kısmın merkezine uzaklığı eşit olan koordinatlar bulunur. Şekil 3.9, dış yarıçapı 10 ve iç yarıçapı 20 olan bir torusu göstermektedir.

Algorithm 6 Torus

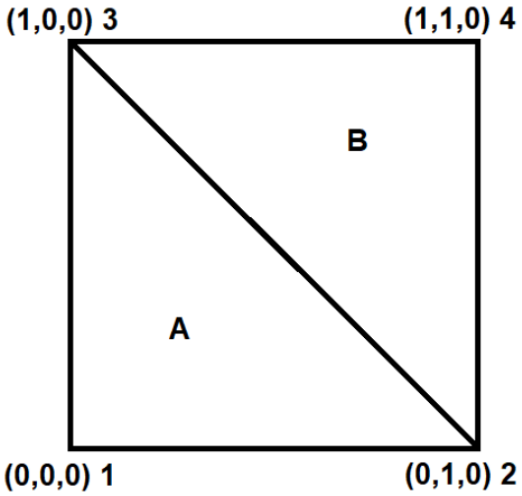
```
0: innerRadius ← radius of the empty space inside Torus
0: outerRadius ← radius of the circular tube of Torus
0: Calculate total radius of Torus.
0: for each points in a box which has dimensions as total radius and outer radius do
0:     Find the centerline position corresponds to the point.
0:     if Distanceofpoint ≤ outerRadius then
0:         Add point to the positions list.
0:     end if
0: end for=0
```

3.3. Voksellerin Oluşturulması

3.3.1. Voksellerin Çizdirilmesi

Bu tez kapsamında oluşturulacak şekiller vokseller ile gösterilecek olsa da, voksellerin ekranda oluşturulması normal polygon işlemesi ile yapılacaktır. Bu işlem ise Unity'nin mesh çizdirme API'si kullanılarak yapılacaktır.

Unity mesh API'si üzerinden bir mesh çizilmesi için gerekli olan asgari bilgiler, mesh'e ait köşe ve üçgenlerin bilgileridir. Köşeler mesh'in köşe noktalarını, üçgenler ise her üç vertisin birbiri ile nasıl birleştiğini gösterir. Köşelerin koordinatları sıralı bir liste olarak tutulurken, üçgen listesi köşelerin sırasına göre tutulur. Üçgen noktalarının sırası köşeler saat yönünde olacak şekilde yazılmalıdır. Aksi takdirde mesh ekrana ters olarak çizdirilir ve istenilen doğrultudan bakıldığında görüntülenmez.



Şekil 3.10 Köşe pozisyonları verilen bir kare mesh'i

Mesh çiziminin daha iyi anlaşılabilmesi için yukarıda verilen şekil örnek olarak kullanılacaktır. Figürde gösterilen kare, iki adet üçgene bölünmüştür. Üçgenler listesine köşelerin sırası saat yönü olacak şekilde eklenecektir. Yukarıdaki örnek ele alınırsa, A üçgeni için 1-3-2, B üçgeni için 2-3-4 sıralaması iki adet doğru yönde üçgen çıkaracaktır. Yani, V=(0,0,0), (0,1,0), (1,0,0), (1,1,0) olarak verilen bir köşe listesine karşılık gelecek üçgen listesi, T=1,3,2,2,3,4 şeklinde olacaktır.

Görüldüğü üzere bir kare mesh'i üretmek için 4 adet köşe yeterlidir. Bir küp yaratılmak istendiğinde ise, köşe listesine toplam 8 tane değil, 24 köşe koyulması gerekmektedir. Bunun sebebi ise Unity Oyun Motoru'nun mesh normallerini köşeler üzerinden hesaplamasıdır. Mesh normalleri, üretilen mesh'ten dışarıya doğru, dik olarak çizilen vektörlerdir. Bu vektörlerin amacı Unity'e ışıklandırma konusunda yardımcı olmaktır. Eğer üretilmek istenen küp mesh'i yalnızca 8 köşe ile çizilirse, şekil ekranda görüntülenecek, ancak ışıklandırma ve gölgelendirmesi yanlış olacaktır. Ancak örnekte gösterilen şekilde çizilen bir kare mesh'i, her bir küp yüzü için farklı oryantasyonlarda 6 kere çizilirse, üretilen mesh'in ışıklandırması çok daha doğru görünecektir. Her yüzün 6 kere çizilmesi ve her yüzde 4 köşe olması sebebiyle bir vokselin çizilebilmesi için en az 24 köşe gerekmektedir.

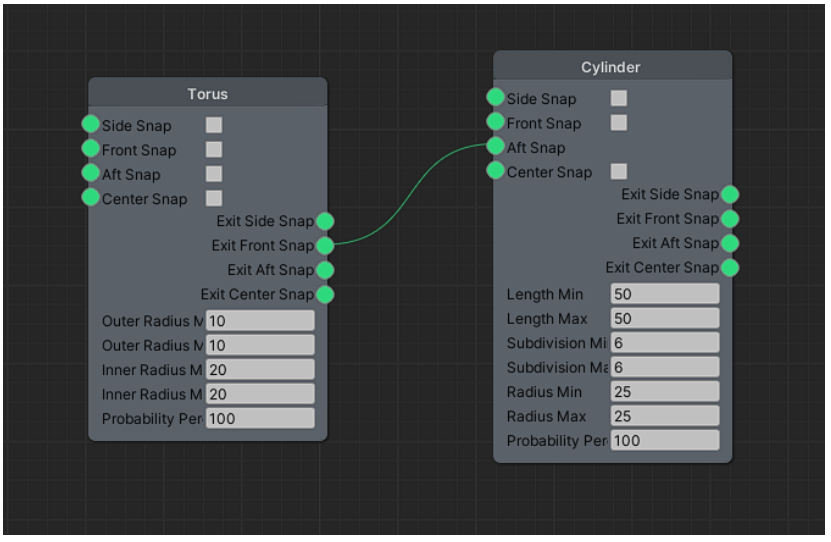
Voksellerin renklendirilmesi için, kullanıcıdan ayrı bir renk parametre nesnesi beklenmektedir. Bu nesne içinde, 4 adet renk ve her bir renk için çıkma ihtimali bulunmaktadır. Vokseller çizilirken, bu renk ve olasılık verileri kullanılarak, her biri rastgele olacak şekilde renklendirilir.

İlkel şekiller oluşturulurken her bir voksel için olan pozisyon bir listede tutulur. Şeklin oluşturulması bittikten sonra bu listedeki pozisyonlara vokseller, yukarıda anlatıldığı gibi yaratılıp yerleştirilir. Bu işlemin sonnunda ilkel şekillerden bir tanesi yaratılmış olur.

3.4. Gramer Kurallarına Göre İlkel Şekillerin Birleştirilmesi

Gramer kurallarına göre şekiller oluşturulurken aralarındaki pozisyon ilişkileri bir veri ağacı içinde tutulur. Böylece her bir şeklin bağlı olması gereken şekillere ulaşım kolaylaşmaktadır. Şekillerin birleştirilmesi bütün şekiller oluşturulduktan sonra başlar.

Şekillerin nasıl birleştirilecekleri, gramer kuralları hazırlanırken düğüm grafiğinde tanımlanır. Her bir düğüm için 3.11 şeklinde gösterilen giriş ve çıkışlar bulunmaktadır.

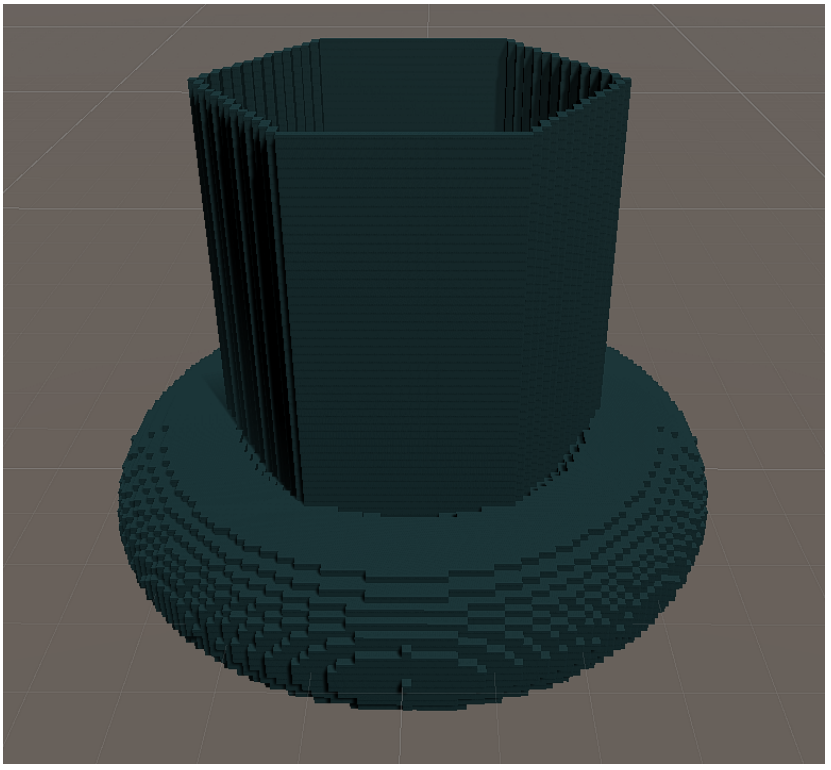


Şekil 3.11 Birbirine bağlı iki düğüm

Her bir şekil oluşturulurken, o şekle ait bağlanma noktaları da tanımlanır. Her şekil için, şeklin tam ortasında merkez bağlanma noktası, şeklin y-ekseninde en üst ve en alt noktaları,

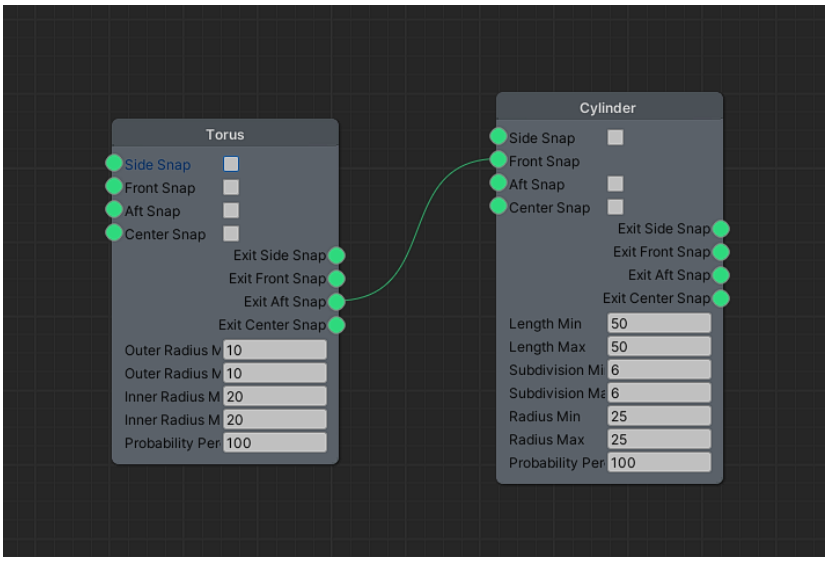
ön ve arka bağlanma noktaları olarak tanımlanırken, şekillerin altbölüm sayısına göre de yan bağlanma noktaları tanımlanır. Örneğin 5 altbölüme sahip olan bir silindir için, bir tane ön bağlanma noktası şeklin en üstünde, bir tane arka bağlanma noktası şeklin en altında, bir tane bağlanma noktası şeklin tam ortasında ve 5 tane yan bağlanma noktası, silindirin her bir altbölüm kenarının ortasında yer almaktadır.

Her şekil diğer şekillerle giriş ve çıkışlarına göre bağlanabilir. Bir şekilden birden fazla çıkış yapılabilmesine rağmen, giriş yalnızca tek bir bağlantıya sahip olabilmektedir. Şekil 3.11 incelenecek olursa, girilen gramer kuralının torus ve silindir yarat, silindiri, silindirin arka bağlanma noktası torusun ön bağlanma noktasına gelecek şekilde kaydır ve birleştir olduğu görülebilir. Şekil 3.12, bu kural setinin ortaya çıkardığı şekli göstermektedir.



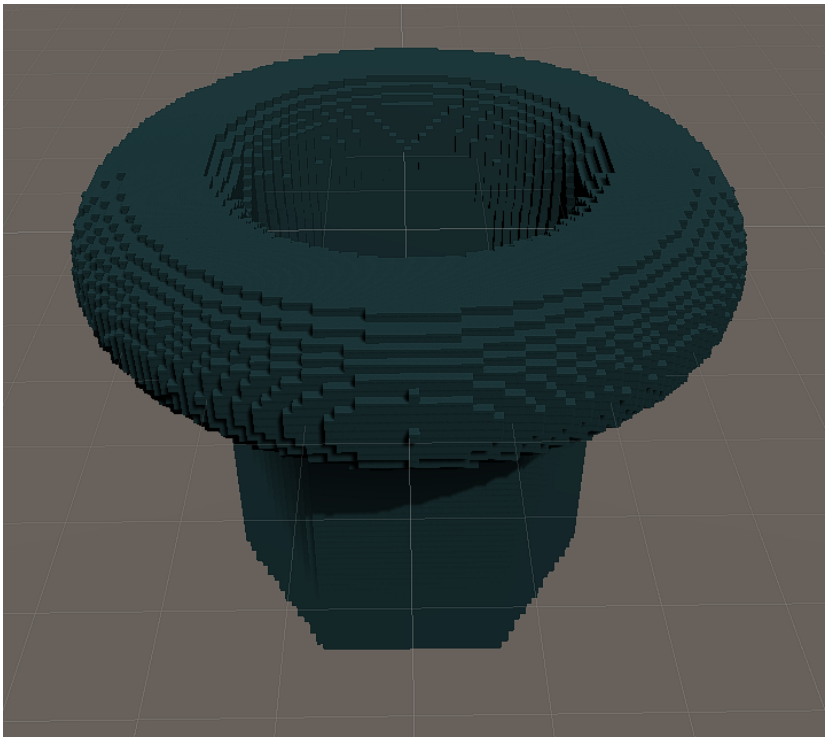
Şekil 3.12 Birbirine bağlanmış iki şekil

Eğer yalnızca bağlantı kuralları Şekil 3.13'de gösterildiği gibi değiştirilirse, torus şeklinin arka bağlanma noktasına, silindirin ön bağlanma noktasının gelecek şekilde bir birleşme oluşması gerektiği görülebilir.



Şekil 3.13 Birbirine bağlı iki düğüm

Bu kurallar kullanıldığında ortaya çıkan model, Şekil 3.14'de görülebilir.



Şekil 3.14 Birbirine bağlanmış iki şekil

4. Sonuçlar

Bu başlık altında, geliştirilen yazılımın deneysel sonuçları anlatılacaktır. Sonuçlar aşağıdaki başlıklar altında incelenecektir:

1. Performans Testi,
2. Anket Sonuçları Analizi,
3. Çeşitlilik Testi,

4.1. Performans Testi

Yazılımın performansı farklı büyüklüklerde ve parça sayıları içeren modellerle test edilecektir. Algoritma, üç boyutlu model oluştururken harcadığı zaman ve en sonda çıkan modelin boyutu üzerinden performansı test edilecektir. Farklı gramer kurallarının yazılımın çalışmasına olan etkisi incelenecektir. Aynı şekil gramerinin her yorumlanması farklı büyüklükte model çıkardığı için, süre ve çıkan modelin boyutu da biraz farklı olabilir. Bu bölüm için 4 adet şekil grameri oluşturulmuş ve testler bu modeller üzerinde yapılmıştır.

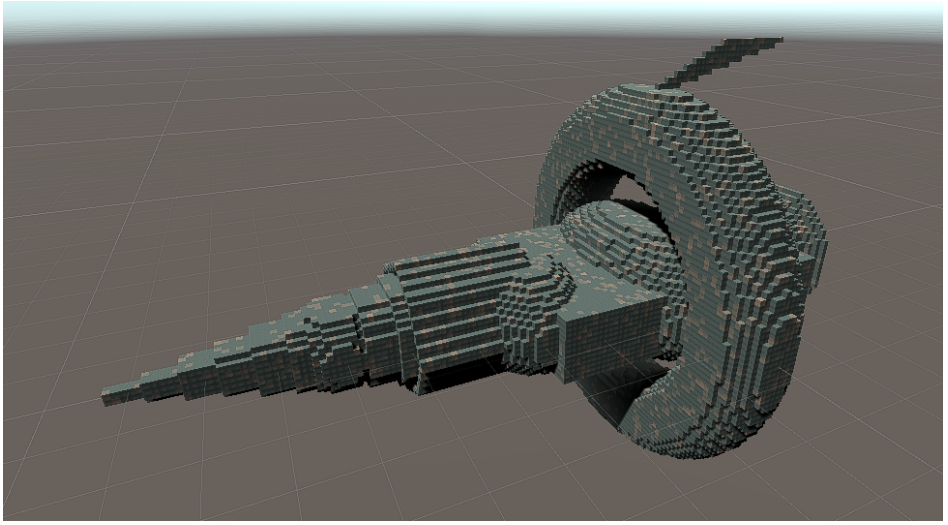
Geliştirilen yazılım, üzerinde Windows 10 çalışan, 16GB RAM, AMD Ryzen 5 2600X işlemci ve NVIDIA GeForce GTX 1660 Ti ekran kartı içeren bir bilgisayarda koşulmuştur.

4.1.1. 1. Model

1. model Şekil 4.1'deki şekil grameri kullanılarak oluşturulmuştur. Bu şekil gramerinin okunmasından model çıkana kadar geçen süre 0.98 - 1.38 saniye arasında değişmiştir. Model oluştuktan sonra Unity Oyun Ekranında 140-200 FPS arası bir kare hızı yakalanmıştır. Bu şekil için dışarıya aktarılan .fbx modelinin boyutunun ise 177 MB olduğu görülmüştür.

Şekil 4.1 1. model için oluşturulan şekil grameri

Şekil 4.1'deki şekil gramerinden çıkan sonuçlardan birisi ise Şekil 4.2'deki gibidir.



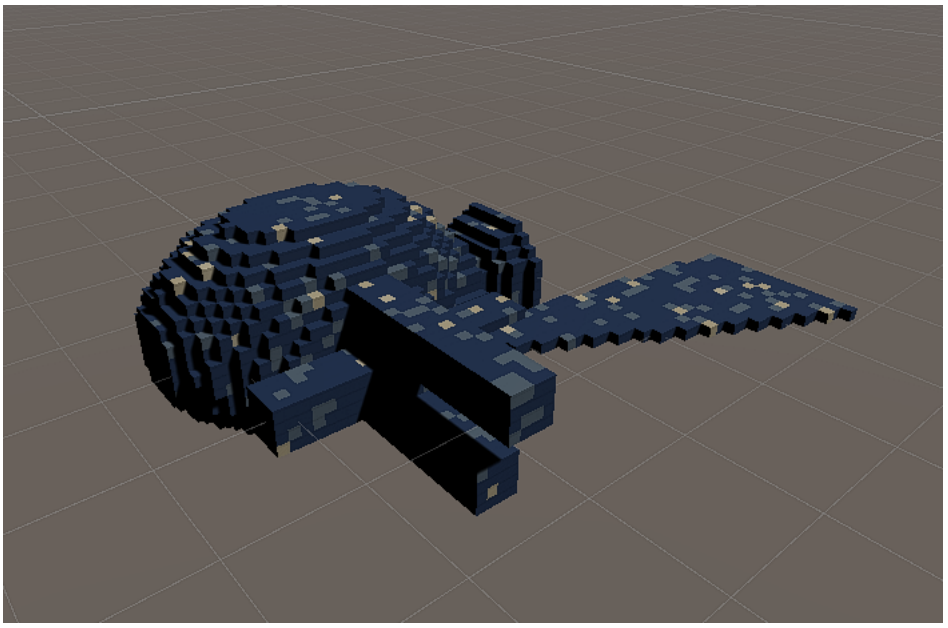
Şekil 4.2 1. model

4.1.2. 2. Model

2. model Şekil 4.3'deki şekil grameri kullanılarak oluşturulmuştur. Bu şekil gramerinin okunmasından model çıkana kadar geçen süre 0.09 - 0.14 saniye arasında değişmiştir. Model oluştuktan sonra Unity Oyun Ekranında 2000-2200 FPS arası bir kare hızı yakalanmıştır. Bu şekil için dışarıya aktarılan .fbx modelinin boyutunun ise 16 MB olduğu görülmüştür. Bu modelde 1. Modele kıyasla daha çok şekil olmasına rağmen, ilkel şekillerin büyüklüklerinin daha küçük olmaları sebebiyle oluşma süresi ve .fbx modelinin boyutu farklı oluşmuştur.

Şekil 4.3 2. model için oluşturulan şekil grameri

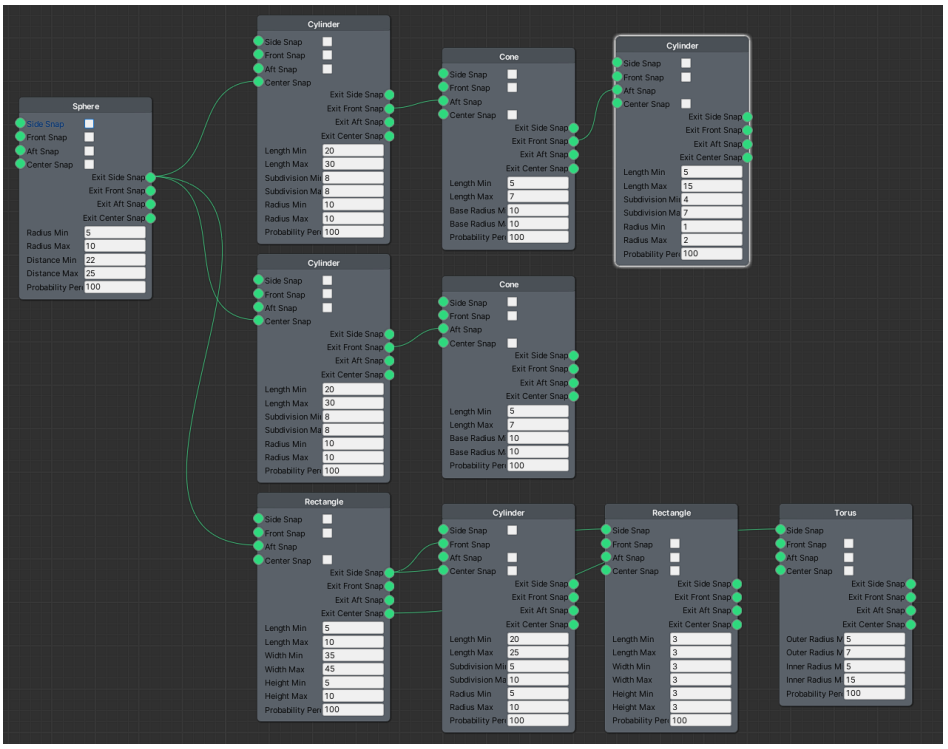
Şekil 4.3'deki şekil gramerinden çıkan sonuçlardan birisi ise Şekil 4.4'deki gibidir.



Şekil 4.4 2. model

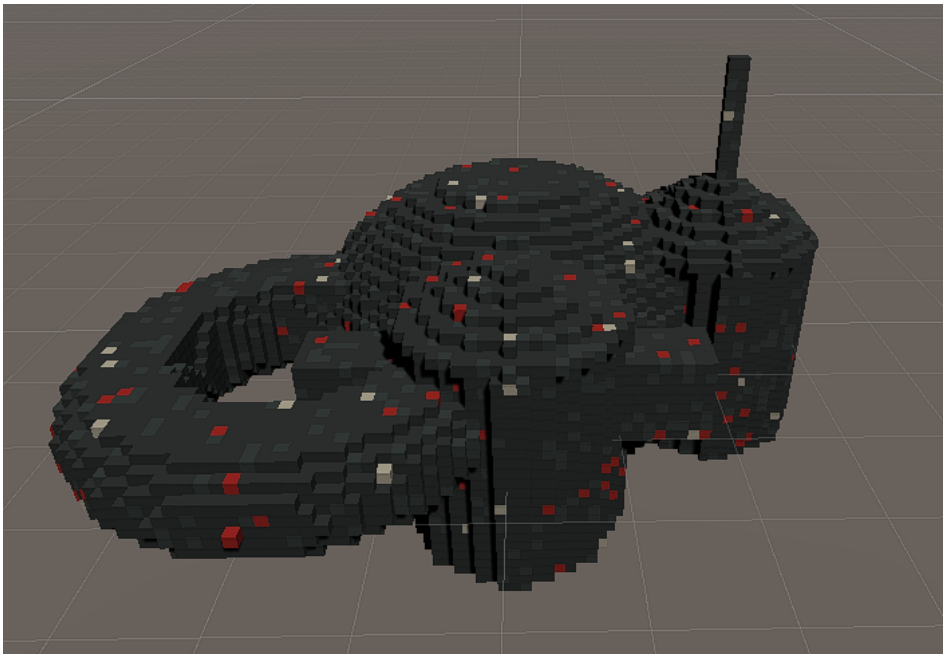
4.1.3. 3. Model

3. model Şekil 4.5'deki şekil grameri kullanılarak oluşturulmuştur. Bu şekil gramerinin okunmasından model çıkana kadar geçen süre 0.41 - 0.59 saniye arasında değişmiştir. Model oluştuktan sonra Unity Oyun Ekranında 2000-2200 FPS arası bir kare hızı yakalanmıştır. Bu şekil için dışarıya aktarılan .fbx modelinin boyutunun ise 93 MB olduğu görülmüştür.



Şekil 4.5 3. model için oluşturulan şekil grameri

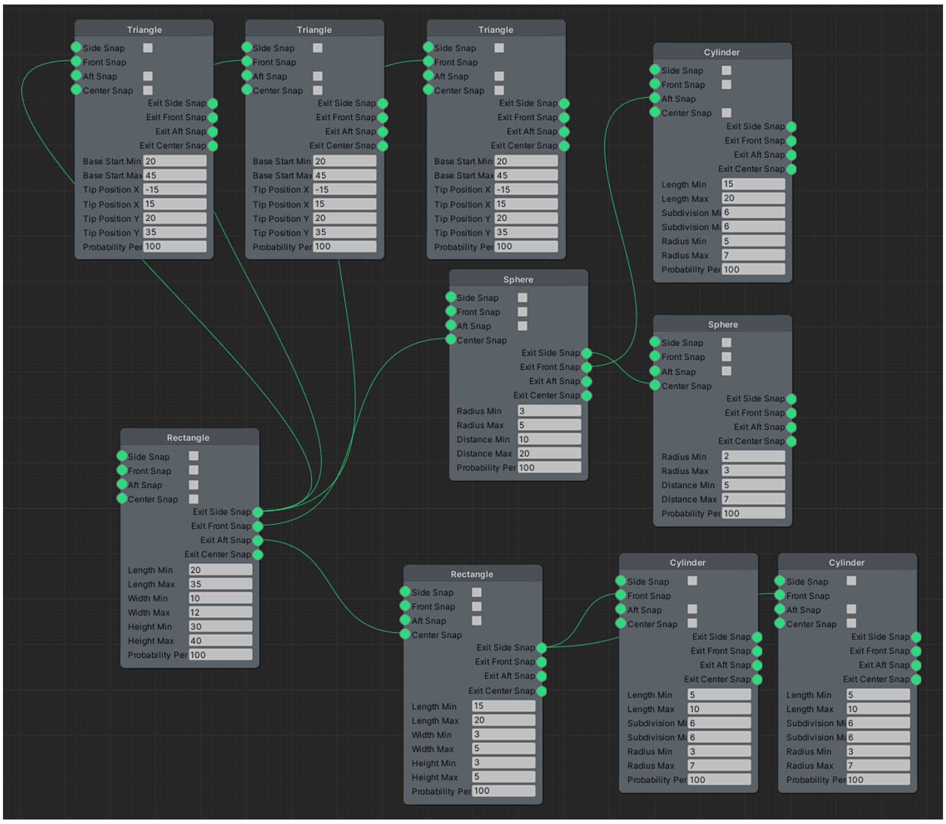
Şekil 4.5'deki şekil gramerinden çıkan sonuçlardan birisi ise Şekil 4.6'deki gibidir.



Şekil 4.6 3. model

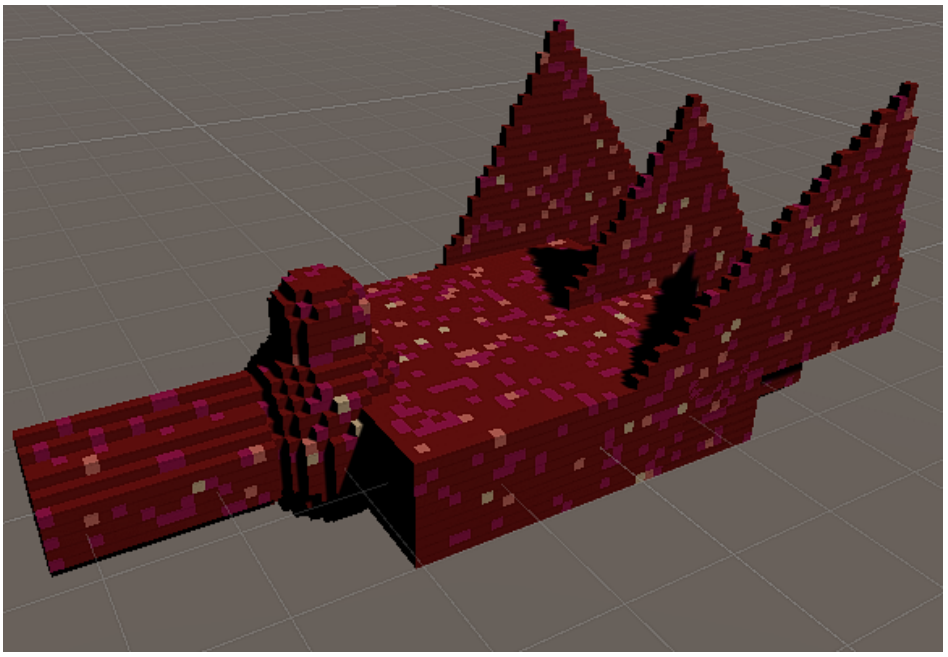
4.1.4. 4. Model

4. model Şekil 4.7'deki şekil grameri kullanılarak oluşturulmuştur. Bu şekil gramerinin okunmasından model çıkana kadar geçen süre 0.21 - 0.24 saniye arasında değişmiştir. Model oluştuktan sonra Unity Oyun Ekranında 2100-2400 FPS arası bir kare hızı yakalanmıştır. Bu şekil için dışarıya aktarılan .fbx modelinin boyutunun ise 28 MB olduğu görülmüştür.



Şekil 4.7 4. model için oluşturulan şekil grameri

Şekil 4.7'deki şekil gramerinden çıkan sonuçlardan birisi ise Şekil 4.8'deki gibidir.



Şekil 4.8 4. model

4.2. Anket Sonuçları Analizi

Bölüm 4.1. Performans Testi başlığı altında listelenen modellerin şekil grameri kuralları kullanılarak elde edilen modellerle bir oyun geliştirilmiştir. Bu oyun daha sonra 20 kişiye oynatılmış ve bir anket ile modelleri değerlendirmeleri istenmiştir. Buna göre oluşan sonuçlar bu bölümde incelenecektir.

Anket kapsamında sorulan sorular aşağıdaki gibidir:

Anket Soruları:

1. Yaş:
2. Cinsiyet:
3. Öğrenim Durumu:
4. Haftada oyun oynanan saat:
5. Daha önce yordamsal üretim kullanan bir oyun oynadınız mı?
6. Yordamsal üretim kullanan oyunlar sizde fazladan bir ilgi oluşturur mu?

Bu anketle birlikte sunulan oyunla ilgili sorular:

7. Oynadığınız oyundaki modellerden ilginizi en çok hangisi çekti?

8. 7. soruda söylediğiniz model neden ilginizi çekti?

9. Oynadığınız oyundaki modeller size yeterince çeşitli geldi mi?

10. Oynadığınız oyundaki modellerin hepsi size bir bakışta uzay araçları olduklarını belli etti mi?

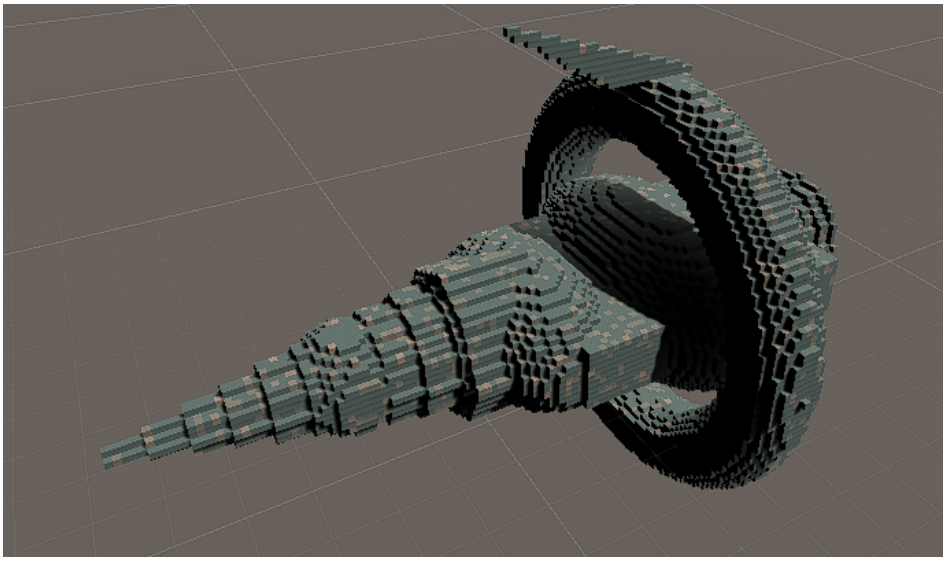
11. Oynadığınız oyundaki uzay araçlarından sadece modeli hoşunuza gittiği için oynadığınız oldu mu?

12. Oynadığınız oyundaki modellerden size yanlış görünen oldu mu?

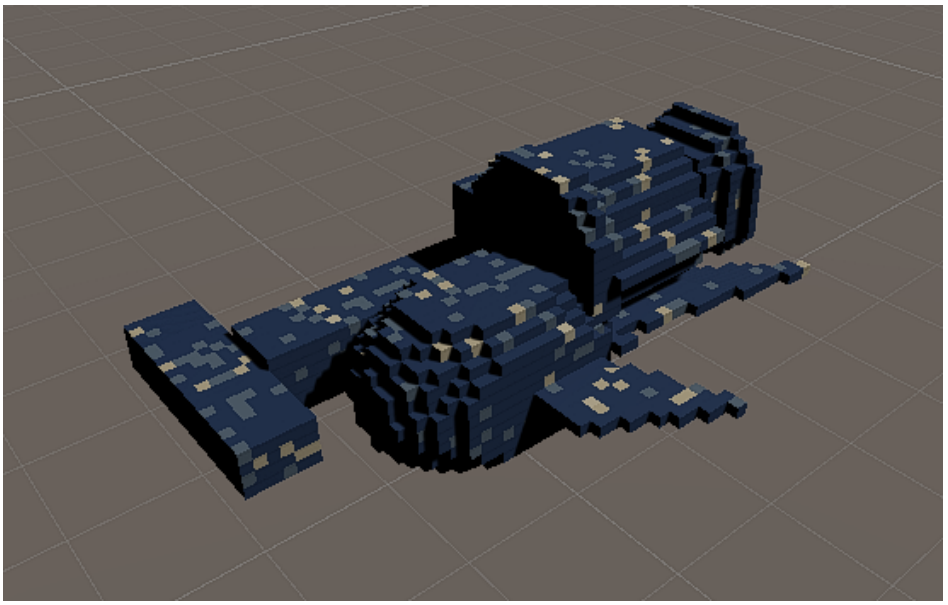
13. 12. soruya yanıtınız evet ise neden?

14. Sizce oynadığınız oyundaki modellerin hepsinin yordamsal olarak oluşturulmuş olmasının oyuna bir katkısı oldu mu?

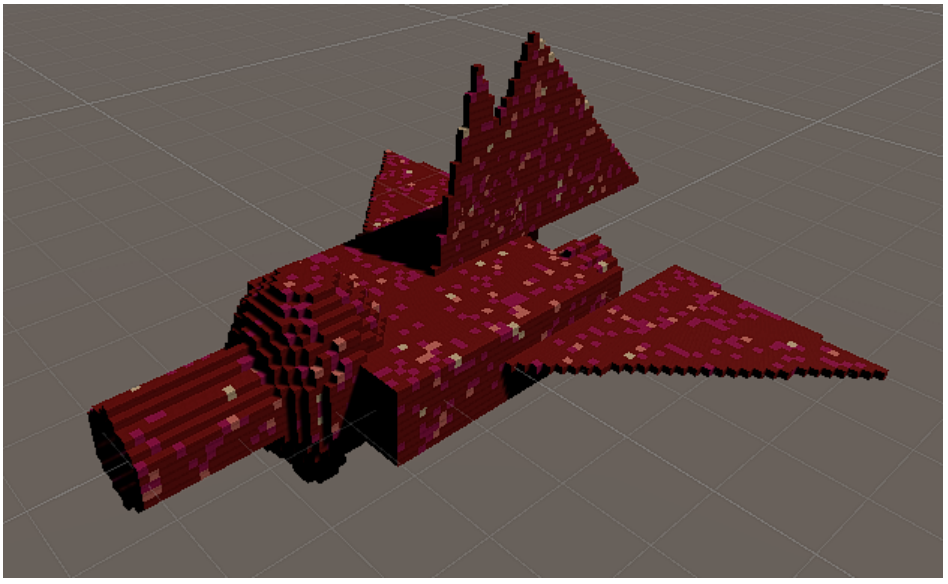
Oyunda kullanılan modeller ise oyundaki isimleriyle Şekil 4.9, Şekil 4.10, Şekil 4.11 ve Şekil 4.12'de verilmiştir.



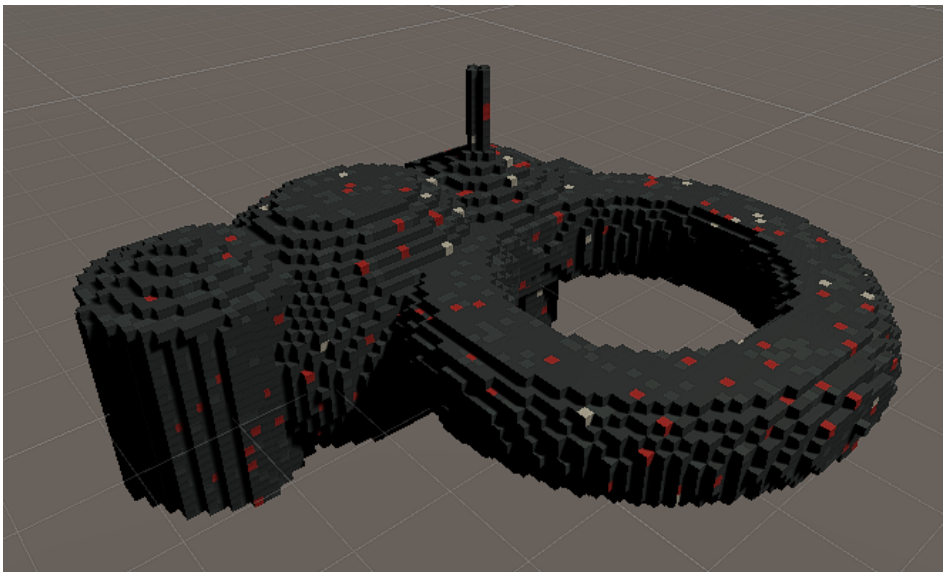
Şekil 4.9 Archer gemisi



Şekil 4.10 Karetta gemisi

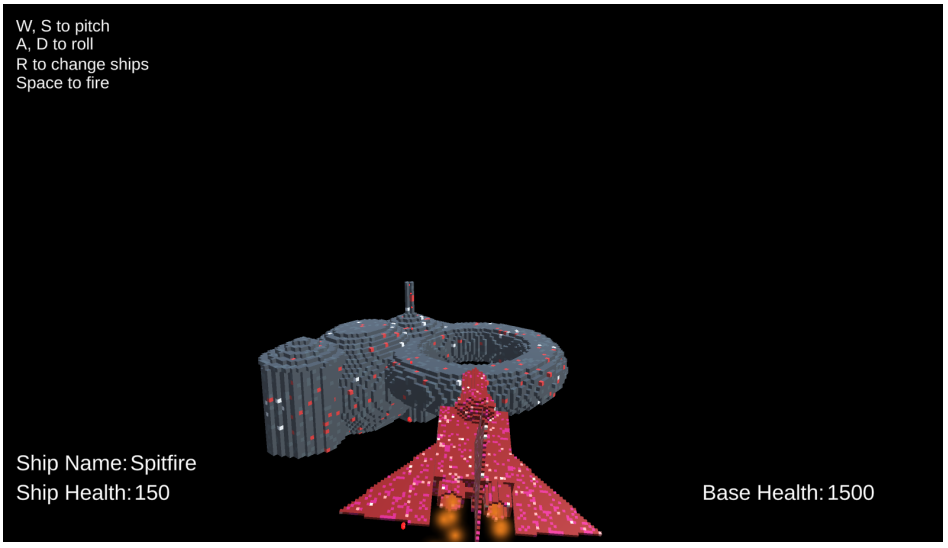


Şekil 4.11 Spitfire gemisi



Şekil 4.12 Base gemisi

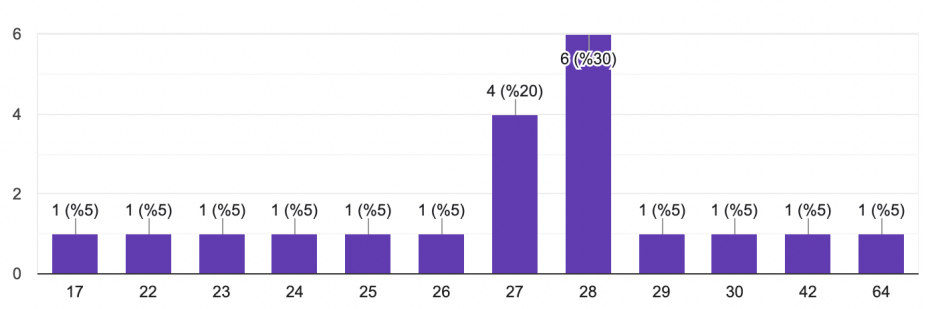
Geliştirilen oyunda, Archer, Karetta veya Spitfire gemilerinden biriyle oynanıp geri ateş eden Base gemisi yok edilmeye çalışılır. Oyuncu istediği zaman gemisini değiştirebilmektedir. Base gemisinden gelen ateşle oyuncunun gemisi yok edilirse oyun bir sonraki gemiye geçerek devam eder. Her geminin kendine ait can, hız ve kullanım kolaylığı değerleri vardır.



Şekil 4.13 Oyunda Spitfire gemisi kullanılırken çekilmiş ekran görüntüsü

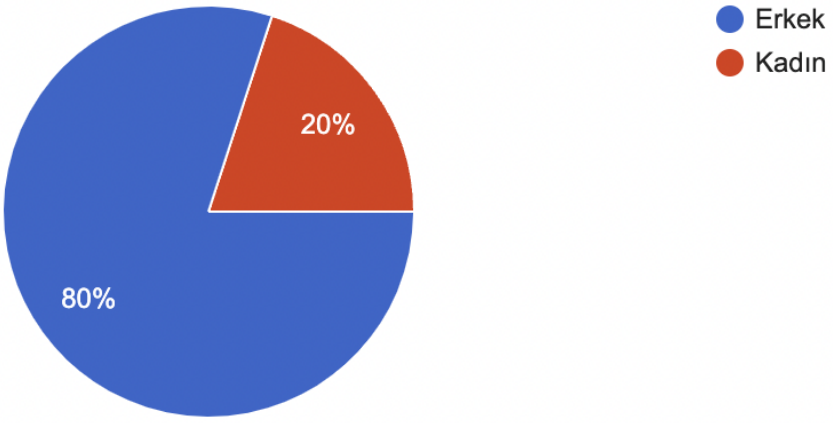
4.2.1. Demografik Sonuçlar

Ankete katılan kişilerin yaş dağılımları Şekil 4.14'de gösterilmiştir.



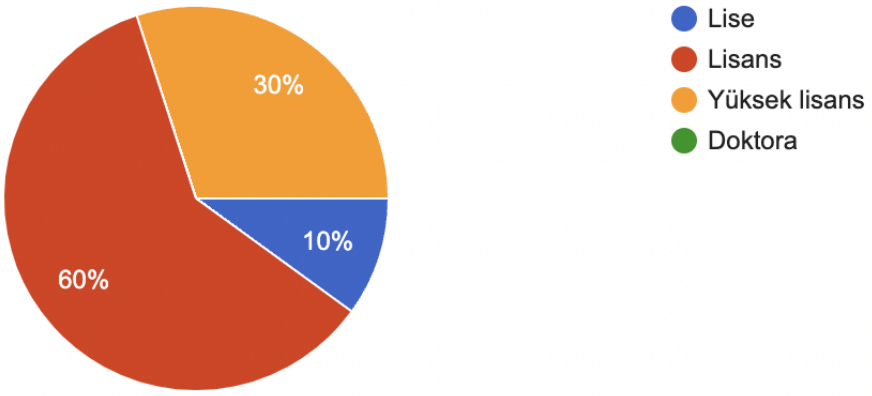
Şekil 4.14 Ankete katılanların yaş dağılımları

Ankete katılan kişilerin cinsiyet dağılımları Şekil 4.15'de gösterilmiştir. Buna göre katılanların 16'sı erkek, 4'ü kadındır.



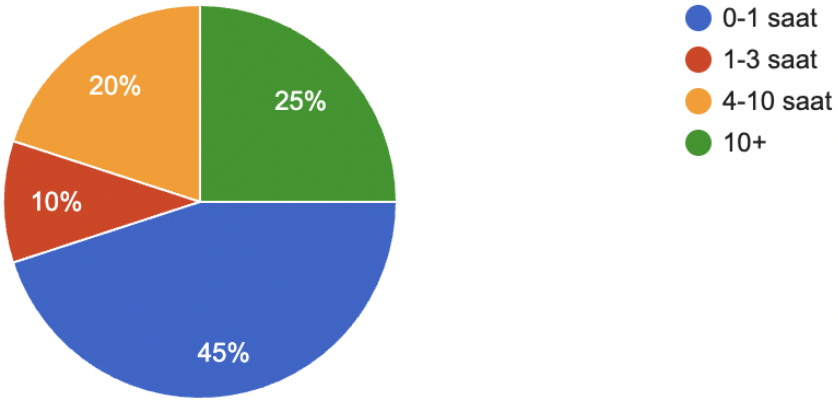
Şekil 4.15 Ankete katılanların cinsiyet dağılımları

Ankete katılan kişilerin öğrenim durumlarının dağılımları Şekil 4.16'de gösterilmiştir. Buna göre ankete katılanlardan 2'si lise, 12'si lisans, 6'sı ise yüksek lisans öğrenimine sahiptir.



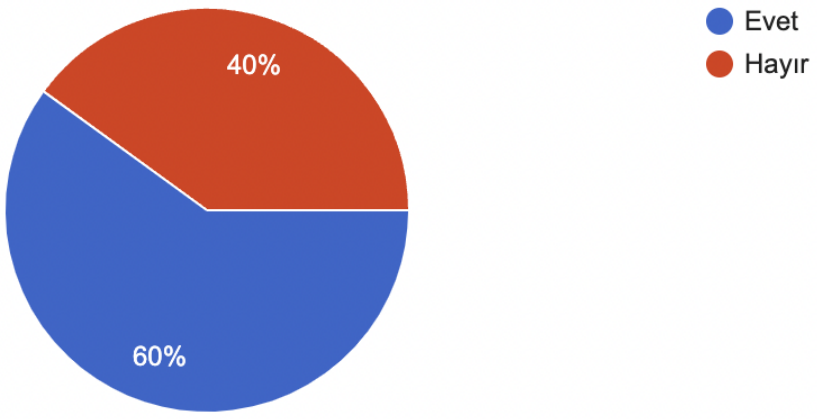
Şekil 4.16 Ankete katılanların öğrenim durumlarının dağılımları

Ankete katılan kişilerin haftada oynadıkları oyun süresi Şekil 4.17'nde gösterilmiştir. Buna göre ankete katılanların 9'u 0-1 saat arası, 2'si 1-3 saat arası, 4'ü 4-10 saat arası ve 5'i 10 saatten fazla bir süreyi haftada oyuna ayırmaktadır.



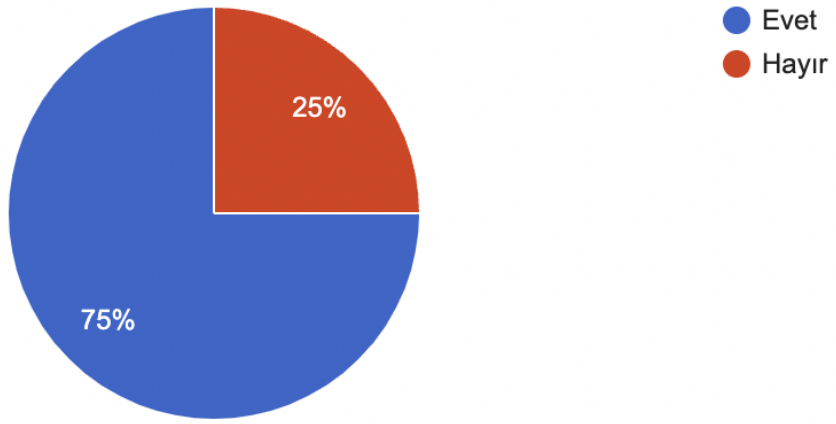
Şekil 4.17 Ankete katılanların oynadıkları oyun süresi

Ankete katılan kişilerin daha önce yordamsal üretim kullanılan bir oyun oynayıp oynamadıklarına verdikleri cevaplar Şekil 4.18'de gösterilmiştir. Buna göre katılımcıların 12'si daha önce yordamsal üretim kullanan bir oyun oynamış, 8'i oynamamıştır.



Şekil 4.18 Ankete katılanların daha önce yordamsal içerik kullanan bir oyun oynama durumu

Ankete katılan kişilerin yordamsal üretim kullanılan oyunlara ilgi duyup duymadıklarına verdikleri cevaplar Şekil 4.19'de gösterilmiştir. buna göre katılımcıların 15'i ilgi duyduğunu, 5'i iseyordamsal içeriklerin onlarda bir ilgi uyandırmadığını söylemiştir.

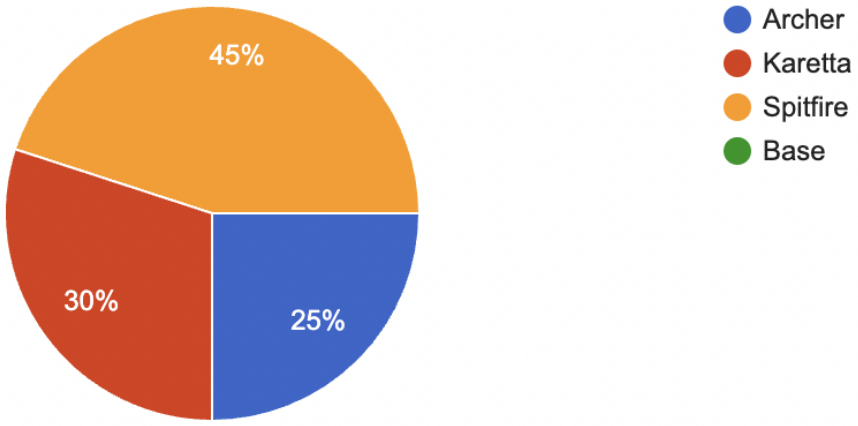


Şekil 4.19 Ankete katılanların yordamsal içeriklere ilgisi

4.2.2. Geliştirilen Oyunla İlgili Sonuçlar

Geliştirilen oyunla ilgili sorulan ilk soru hangi geminin katılımcının ilgisini çektiği ve nedenidir. Bu soruya verilen cevaplar Şekil 4.20'de görülebilir. Buna göre 9 kişi Spitfire

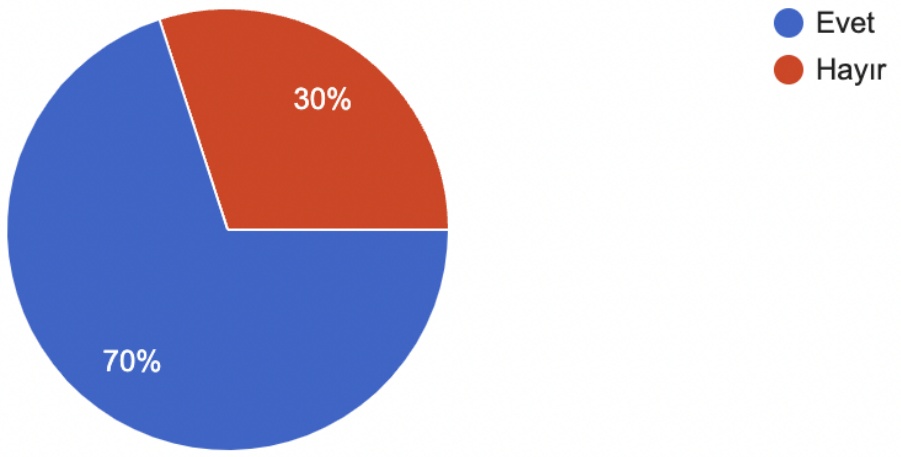
gemisini, 6 kişi Karetta gemisini ve 5 kişi Archer gemisini ilgi çekici bulmuşlardır. Base gemisini ise ilgi çekici bulan birisi olmamıştır.



Şekil 4.20 Ankete katılanların ilgisini çeken gemiler

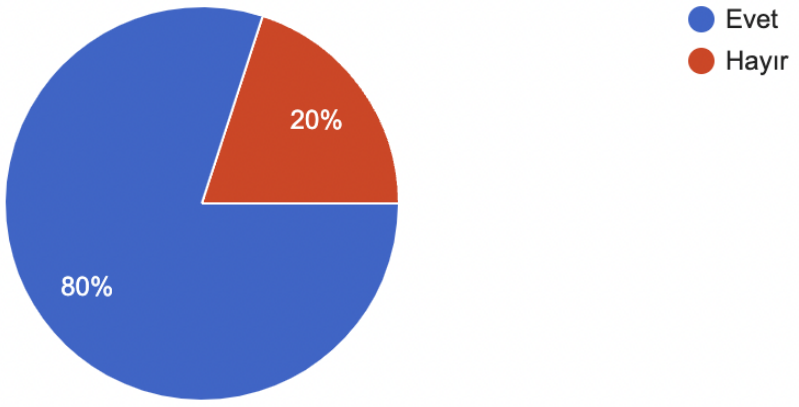
Bu sonuca göre Şekil 4.12'de görülen Base gemisi kimsenin ilgisini çekmemiş, en çok ilgiyi ise Şekil 4.11'de görülen Spitfire gemisi çekmiştir. Katılımcılara seçimlerinin nedenleri sorulduğunda farklı cevaplar alınmıştır. Archer gemisini seçen katılımcılar, geminin orijnal bir tasarım olup, bir amiral gemisini andırdığını ve bir uzay aracına en çok benzeyenin o olduğunu belirtmiştir. Karetta gemisini seçen katılımcılar arasında bir ayrım olduğu görülmektedir. Bir yanda küçük ve sevimli geldiği için bu seçeneği işaretleyenler varken diğer yanda tasarımını çok kötü buldukları için ilgisini çektiğini söyleyenler de olmuştur. Son olarak en çok ilgi çeken gemi olan Spitfire gemisi gelmektedir. Bu gemiye olan yoğun ilginin en büyük nedeninin tanıdık bir model olması olduğu görülmüştür. Klasik bir avcı uçağına benzediği için katılımcılardan daha büyük bir ilgi görmüştür. Ayrıca, bazı katılımcılar bu modelin, yordamsal üretimin daha estetik ve kullanışlı modeller oluşturmada kullanılabileceğini ve endüstride de yeri olabileceğini önizlediği için ilgi çektiğini belirtmiştir.

Katılımcılara geliştirilen oyunda bulunan modelleri çeşitli bulup bulmadıkları sorulmuş ve Şekil 4.21'de görülen cevaplar alınmıştır. Buna göre, 14 kişi modelleri çeşitli bulurken, 6 kişi için model çeşitliliği yeterli gelmemiştir.



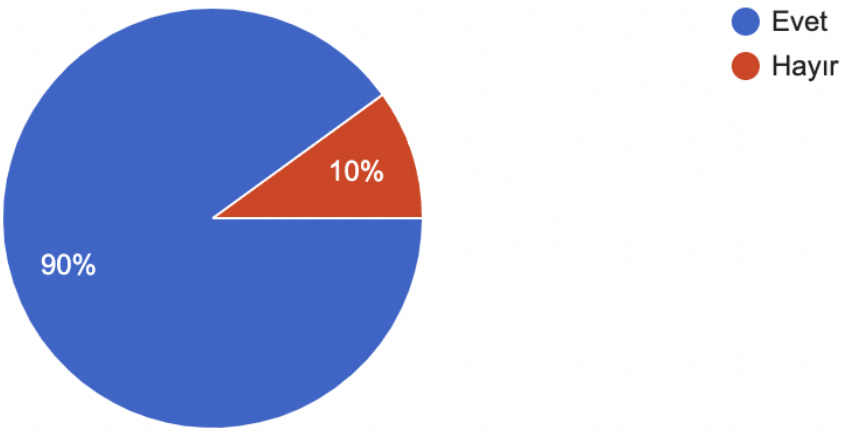
Şekil 4.21 Ankete katılanların modelleri çeşitli bulma durumu

Ankete katılanlara sorulan bir başka soru ise, modellerin bir bakışta uzay gemisi olup olmadığının anlaşılabilirliğiydi. Şekil 4.22'de bu soruya verilen cevapların yüzdeleri görülebilir. Buna göre, 16 kişi modellerin bir bakışta uzay aracı olduklarının anlaşıldığını, 4 kişi ise anlaşılamadığını belirtmiştir.



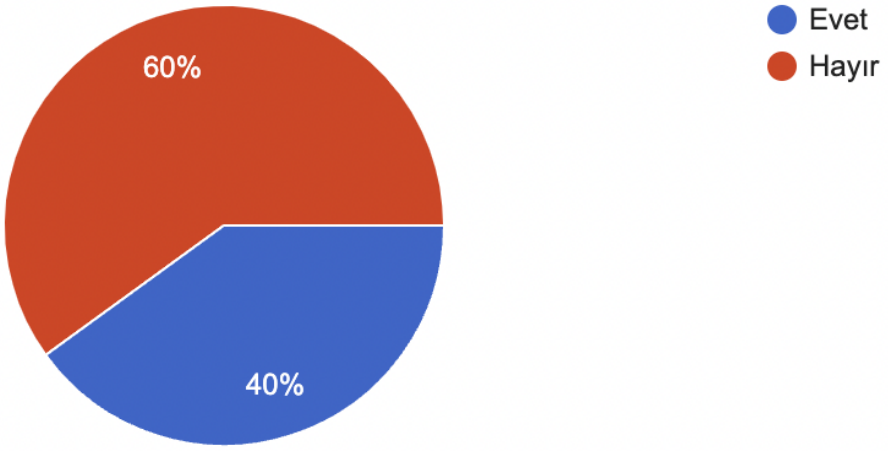
Şekil 4.22 Ankete katılanların modellerin uzay aracı olduğunu anlama yüzdesi

Bir sonraki soruda kullanıcılara, bir modeli yalnızca hoşlarına gittiği için oynayıp oynamadıkları soruldu. Şekil 4.23'de katılımcıların bu soruya verdikleri cevaplar görülmektedir. Buna göre, 18 kişi yalnızca model hoşlarına gittiği için o modeli oynadığını söylerken, 2 kişi yalnızca hoşuna gittiği için bir modeli oynamadığını söylemiştir.



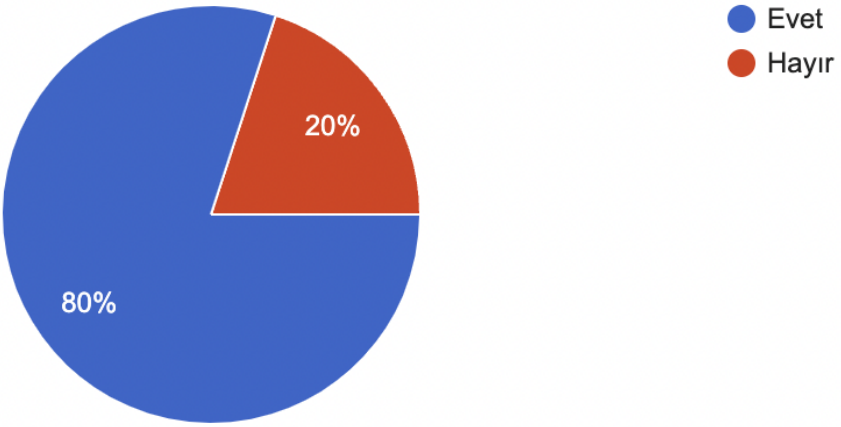
Şekil 4.23 Ankete katılanların bir modeli hoşuna gittiği için oynama yüzdesi

Ankete katılanlara oynanan oyunda yanlış göründüğünü düşündükleri bir model olup olmadığı ve nedeni sorulmuştur. Bu soruya verilen cevaplar Şekil 4.24'de görülebilir. Buna göre, 12 kişi hayır cevabını verirken, 8 kişi yanlış göründüğünü düşündükleri modellerin olduğunu söylemiştir. Verilen cevaplara göre en çok Şekil 4.10'de görülen Karetta gemisinin yanlış göründüğü düşünülmüştür. Geminin görüntüsü alışılagelmiş uzay gemisi şekillerinden farklılık gösterdiği için bazı katılımcılar doğru görünmediğini düşünmüştür.



Şekil 4.24 Ankete katılanların yanlış göründüğünü düşündüğü bir model olup olmadığının yüzdesi

Sonuncu soruda ise modellerin yordamsal olarak üretilmesinin oyuna bir katkısı olup olmadığı sorulmuştur. Şekil 4.25'de bu sorunun cevabı görüntülenmektedir. Buna göre, 16 kişi modellerin yordamsal olarak yaratılmasının oyuna katkı sunduğunu belirtirken, 4 kişi aksini düşünmüştür.



Şekil 4.25 Ankete katılanların yordamsal modellerin oyuna katkısı yorumları

Bu anket 17-64 yaşları arasında, farklı öğrenim durumlarındaki çok karma bir toplulukla yapılmıştır. Sonuç olarak bu çalışma kapsamında geliştirilen yazılımla üretilen üç boyutlu modeller bir oyun içinde kullanıldığında genel olarak pozitif bir tepki almıştır. Çıkan

başka bir sonuç ise insanların genelde daha tanıdık modellere kendilerini yakın hissetmeleri olmuştur. Gerçekçi bir avcı uçağına benzeyen Spitfire gemisi çok beğenilip ilgi görürken, daha alışılmışın dışında bir model olan Karetta gemisi sevilmeyen gemiler arasında ilk sıradadır. Buna rağmen, modellerin yordamsal olarak oluşturulması, modelleme bilmeyen birisi için oyunun geliştirilme sürecini oldukça hızlandırmış ve oyunculardan çoğunlukla olumlu tepkiler alınmıştır.

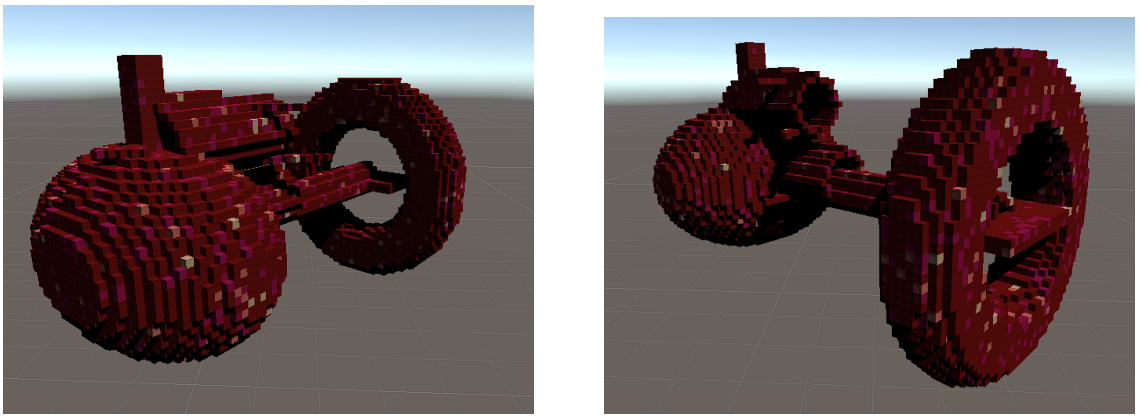
4.3. Çeşitlilik Testi

Bu bölümde bir gramer kuralı setinden ne kadar çeşitli model oluşturulabildiği incelenecektir. İlk olarak oluşturulan şekil grameri seti Şekil 4.26'de verilmiştir.

Şekil 4.26 Çeşitlilik testi için oluşturulan ilk gramer kuralı

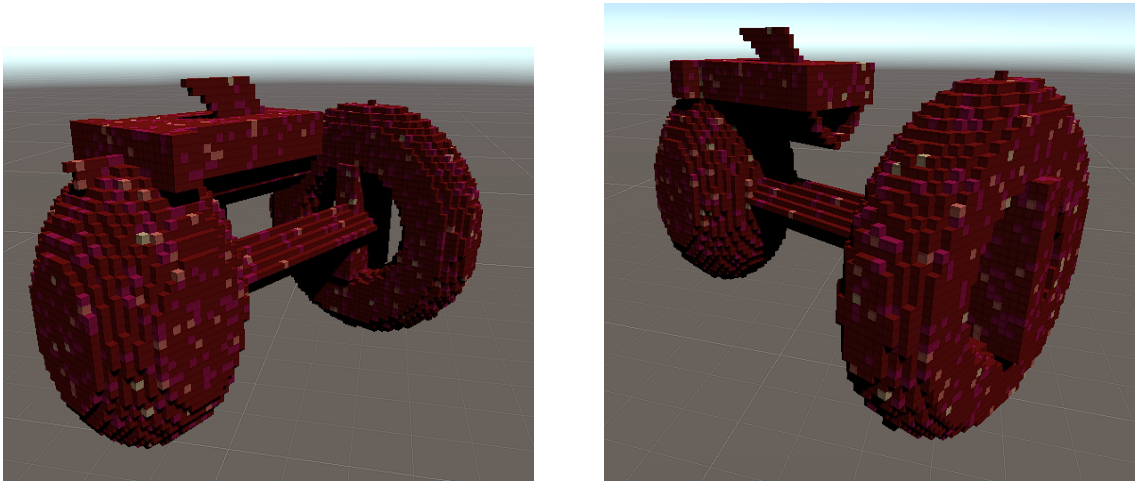
Eğer kural seti incelenirse, iki tane şeklin (bir küre ve bir silindir) oluşma olasılıklarının %70 olarak belirlendiği görülebilir. Eğer kural ağacı üzerindeki bir şekil oluşmazsa, ona bağlı olan şekiller de oluşmaz. Bu sayede yalnızca şekillerin pozisyonu ve büyüklükleriyle değil, şekillerin oluşma ihtimalleri ile de çeşitli şekiller oluşturulabilir. Şekil 4.27, 4.28, 4.29, 4.30 bu gramer kuralları kullanılarak oluşan farklı şekilleri göstermektedir. Şekiller üzerinde hem

ayrıntısal hem de yapısal bir çok farklılık olmasına rağmen birbirlerine oldukça benzedikleri, ve aynı kültürün ürünü oldukları söylenebilir. Yalnızca Şekil 4.29'de diğer modellerde bulunan torus ve küreyi içeren yapı bulunmamaktadır. Bu durumda oluşan şekil çok farklı olsa da, diğer modellerin üst yapısına benzer bir model oluşmuştur. Modelin kullanılacağı medyanın senaryo veya hikayesine göre daha büyük bir gemiye bağlabilen küçük bir gemi olarak yorumlanabilir.



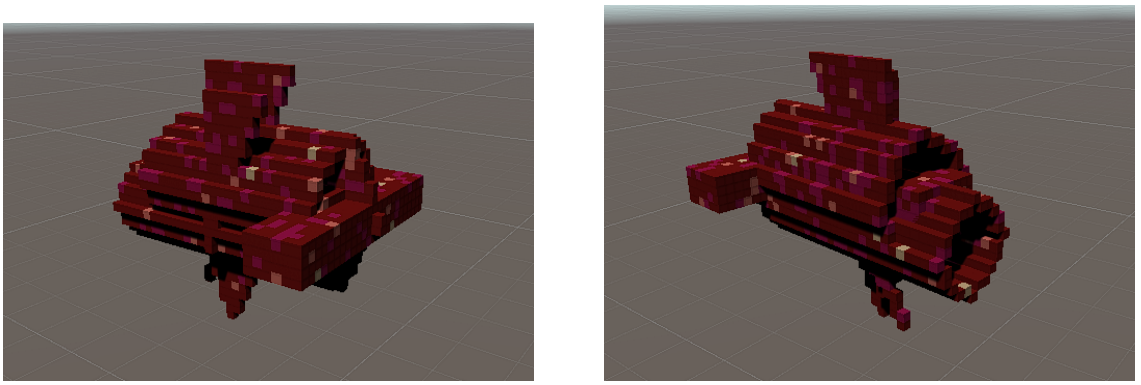
(a) (b)

Şekil 4.27 İlgili kurallar için bir varyasyon



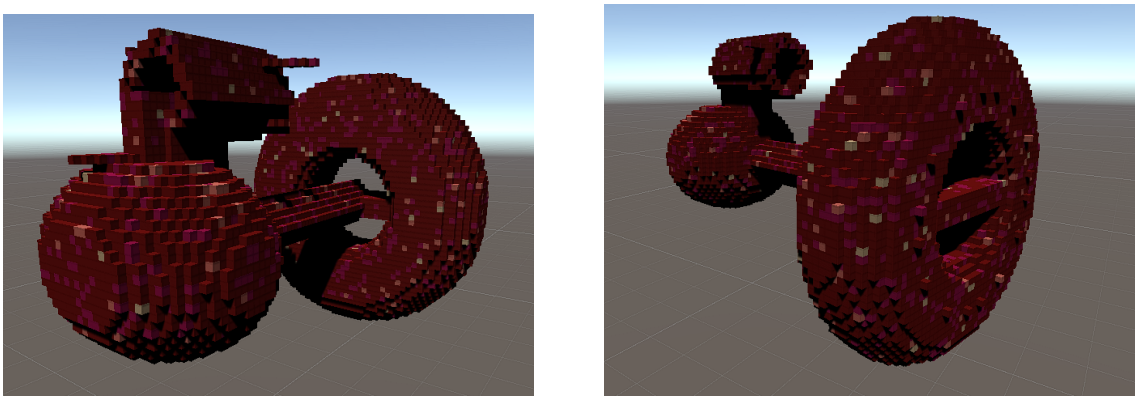
(a) (b)

Şekil 4.28 İlgili kurallar için bir varyasyon



(a) (b)

Şekil 4.29 İlgili kurallar için bir varyasyon



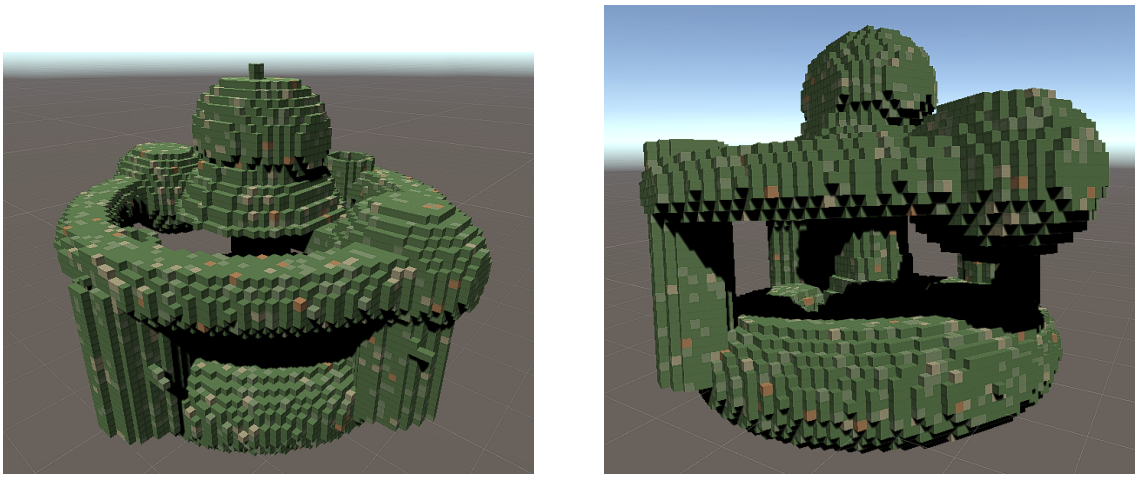
(a) (b)

Şekil 4.30 İlgili kurallar için bir varyasyon

Başka modeller oluşturmak için bir başka şekil grameri seti Şekil 4.31'de verilmiştir.

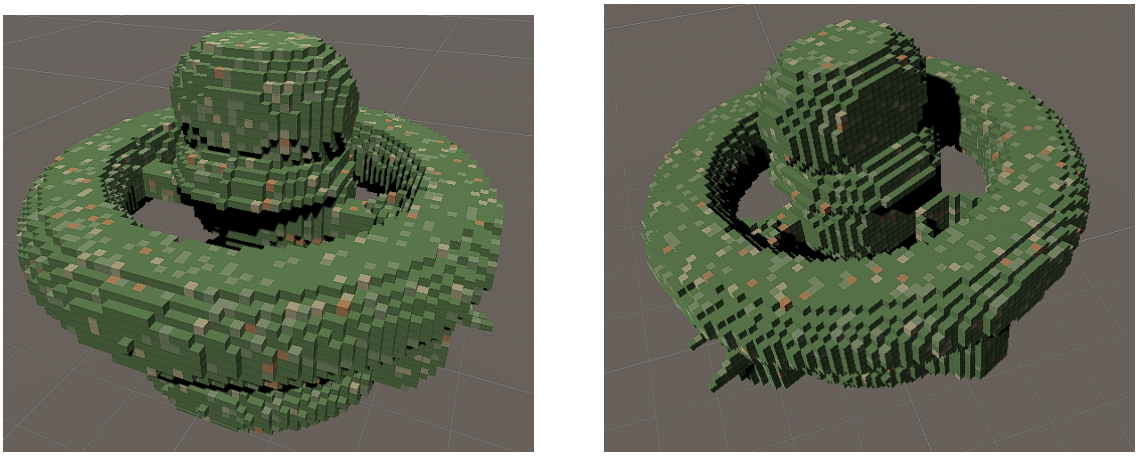
Şekil 4.31 Çeşitlilik testi için oluşturulan ikinci gramer kuralı

Şekil 4.31'deki gramer kuralı incelenirse, bu kural setinde de çıkmama ihtimali olan şekiller görülebilir. Şekil 4.32, 4.33, 4.34, 4.35 bu gramer kuralları kullanılarak oluşan farklı şekilleri göstermektedir. İhtimaller dahilinde modele eklenmeyen parçalar, Şekil 4.34 ve Şekil 4.35 karşılaştırılırsa görülebilir. İki torus arasındaki destek silindirlerinden birisi ve ortadaki yapı ve üstteki torus arasındaki bağlantı parçası Şekil 4.35'te görülmemektedir.



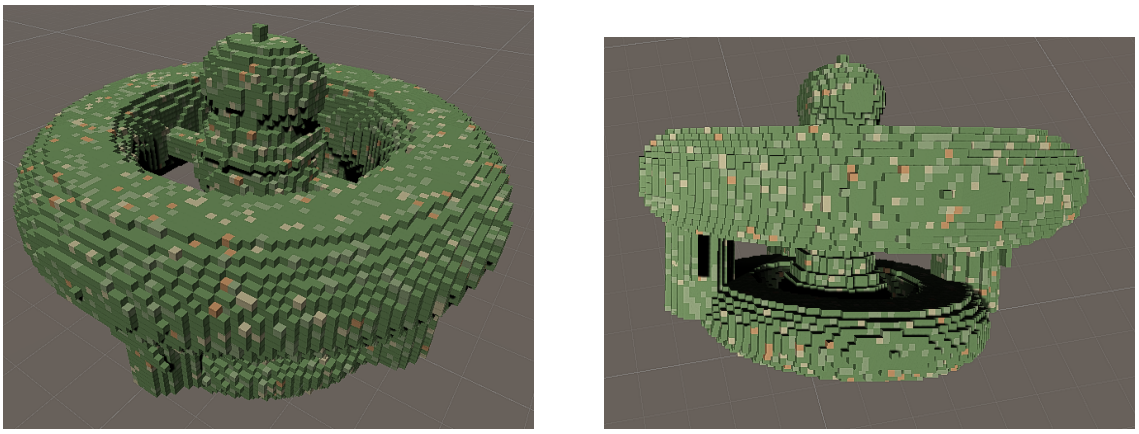
(a) (b)

Şekil 4.32 İlgili kurallar için bir varyasyon



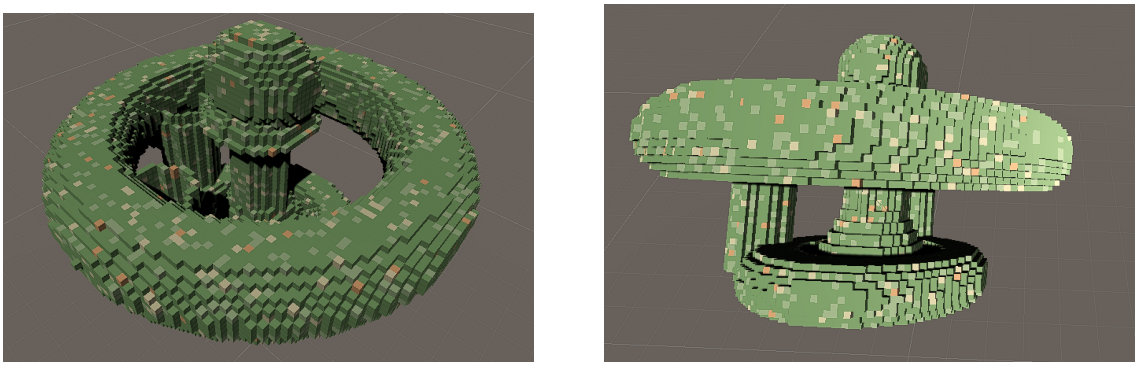
(a) (b)

Şekil 4.33 İlgili kurallar için bir varyasyon



(a) (b)

Şekil 4.34 İlgili kurallar için bir varyasyon



(a) (b)

Şekil 4.35 İlgili kurallar için bir varyasyon

Görüldüğü üzere, aynı şekil grameri kullanılan modellerde birbirlerine benzerlik oranı yüksektir. Tabi ki bu oran yine kullanıcının oluşturduğu şekil gramerine bağlıdır. Eğer şekil grameri hazırlanırken bütün min-max değerleri kendi içlerinde eşit ve bütün oluşma ihtimalleri %100 olarak verilirse her seferinde aynı şekil ortaya çıkacaktır. Böylece kullanıcıya ortaya çıkan model üzerindeki bütün rastgeleliği kaldırma imkanı da verilmiş olmaktadır. Ancak bütün değerler çok farklı verilirse de bu sefer anlamsız şekiller çıkma ihtimali de oluşabilir. Grameri hazırlayan kişi, şekillerin birbiriyle ilişkilerini göz önünde bulundurarak gramer kurallarını oluşturmalıdır.

Aynı şekil grameri kullanan modellerde benzerlik oranı yüksek olsa da farklılıklar da göze çarpmaktadır. Ancak bu farklılıkların çok uç noktalarda olmaması, çıkan modelin stilinin ve çıkış noktasının tutarlı olmasını sağlamaktadır. Bu durum da modellerin tanınabilirliklerinin yüksek olmasına yol açar. Örneğin Şekil 4.31'deki şekil gramerinin çıktılarının bir avcı gemisi değil de, bir uzay üssü olabileceği düşüncesi daha ağır basmaktadır.

Farklı gramer kuralları verildiğindeki çeşitliliğe bakılacak olursa, bu bölümde incelenen modellere ek olarak Bölüm 4.1. Performans Testi ve Bölüm 4.2. Anket Sonuçları Analizi bölümlerinde gösterilen modeller de incelenebilir. Şimdiye kadar görülen modellerin her birinin kendine özgü ve farklı karakterlerde modeller oldukları görülebilir. Bu da şekil gramerlerinin güçlü olduğu noktayı bize göstermektedir. Bir kural dizisi tanımlanırken iki şeklin bağlanma sırası bile değiştirilirse sonuçta ortaya çıkan şekil tamamen farklı olabilmektedir.

Modellerin çeşitliliğinin kaynağı yalnızca şekil grameri kuralları ve algoritma değil, kullanıcının son modeli manipüle edebilmesi ile de sağlanabilir. Oluşturulan modeller Unity Oyun Motoru üzerinde birer mesh objesi olarak yaratılmaktadır. Her bir ilkel şekil de ayrı bir obje ve ona bağlı olan şekiller de o objenin çocukları olarak oluşturulmaktadır. Böylece kullanıcı çıkan şekli ve içindeki alt şekilleri istediği gibi çevirip hareket ettirebilir. Bir alt şekil hareket ettirildiğinde ona bağlı olan şekillerin hepsi de aynı şekilde hareket edeceği için bütünlük daha sağlıklı bir şekilde korunur. Böylece kullanıcı modele son dokunuşu kendisi de yapabilir.

5. Sonuç

Yordamsal üretim, oyun dünyasında çok uzun süreden beri var olan bir yöntemdir. Arazi oluşturmaktan, üç boyutlu modeller oluşturmaya kadar geniş bir kullanım yelpazesi vardır. Bu içeriklerin oyun dosyalarında yer kaplamak yerine oyun oynandığı sırada oluşturuluyor olması, içerik bakımından en zengin oyunların bile boyutlarının küçük olmasını sağlamaktadır. Bu durum hem geliştiricileri büyük bir modelleme iş yükünden kurtarmakta, hem de oyunun her seferinde farklı bir deneyim sunarak devamlılığını arttırmaktadır. Parametrik üretim ise kullanıcının oluşan içerik üzerinde daha büyük bir kontrolünün olmasını sağlayan bir yöntemdir. Yordamsal üretimde, kullanıcıdan çok az veya hiç girdi alınmadan içerik üretilirken, parametrik üretimde üretilecek modelin boyutları ve farklı parametreleri kullanılarak içerik oluşturulmaktadır.

Bu tez kapsamında üç boyutlu modeller oluşturabilmek için bir parametrik içerik üretici geliştirilmiştir. Bu parametrik üretici şekil gramerlerini kullanarak, birbiriyle ilişkili olabilecek ama yine de farklı görünen modeller yaratılmasını sağlamıştır. Bu modeller, basit ilkel şekiller birbirlerine birleştirilerek oluşturulmaktadır. Şekil gramerleri hem ilkel şekillerin hangi parametrelerle oluşturulacağını hem de nasıl birleştirileceğini belirtir. Bu basit şekiller birleştirildiğinde çok daha karmaşık şekillerin ortaya çıktığı görülmüştür.

Çalışma dünydaki en popüler oyun motorlarından biri olan Unity Oyun Motorunda geliştirilmiştir. Geliştirici bir model oluşturmak istediğinde, bir düğüm grafiği yardımıyla, istediği şekilleri, istediği parametrelerle birbirine bağlar ve yazılımı çalıştırır. Yazılım öncelikle düğüm grafiğindeki oluşturulacak bütün şekilleri bir ağaca dizer ve daha sonra her bir ilkel şekli voksellerden oluşacak şekilde yaratır. Daha sonra bu şekiller, şekil gramerindeki düğümlerin ilişkilerine göre manipüle edilir ve ortaya son model çıkar. Geliştirici istediği takdirde bu modeli bir .fbx dosyası halinde çıkarabilir.

Bu yazılımın geliştiricilere en büyük faydalarından birisi, görsel bir arayüzü olmasıdır. Yapılan çalışmalara bakıldığında genellikle şekil gramerlerinin kodlama tarzı arayüzlere

sahip olduğu görülmektedir. Ancak geliştirilen yazılımın düğüm grafikli arayüzü daha erişilebilir bir kontrol katmanı sağlamaktadır.

Bu araştırma sonucunda çıkan modellere oyuncuların verecekleri tepkilerin ölçülmesi için, birkaç tane model oluşturulmuş ve bu modellerle bir oyun geliştirilmiştir. Bu oyunu oynayan insanlara modellerin çeşitliliği, ilgi çekiciliği gibi konularda anket yapılmıştır. Bu anketin sonuçlarına göre, oyuncular çıkan çoğu şekli özgün ve ilginç bulduklarını belirtmişlerdir. Bu oyunun kısa bir sürede ve modelleme tecrübesi olmadan geliştirildiği düşünülürse, yazılımın, oyun geliştiricileri için de kullanışlı bir araç olabileceği görülmüştür.

Bu tez kapsamında yapılmış çalışmanın ilerlemeye ve geliştirilmeye açık yanları vardır. Öncelikle yapılabilecek bir geliştirme modellerin dokularının iyileştirilmesidir. Yazılımın şuanki halinde dokular, verilen renk paletine göre vokseller rastgele boyanarak oluşturulmaktadır. Doku ataması buraadan iki yönde iyileştirilebilir. Birincisi, voksellere renkler atanırken, verilen renk paletindeki renklerin daha uyumlu bir uyumlu bir şekilde dağıtılmasıdır. Bir voksel boyanırken etrafındaki vokseller de incelenip ona göre bir renk seçilebilir. Bu da model üzerindeki renk geçişlerinin göze daha iyi görünecek şekilde oluşmasını sağlayabilir. Bir diğer doku geliştirmesi ise dokunun modele elle oluşturulup atanmasını sağlayacak bir sistem geliştirilebilir. Bu sistem, modeli açarak iki boyutlu haline getiren bir algoritma içermelidir. Daha sonra doku sanatçıları bu iki boyutlu model üzerinde istedikleri dokuyu hazırlayıp tekrar üç boyutlu modele geçiş yaptırabilirler. Bu şekilde model yordamsal olarak yaratılır ancak üzerindeki doku tamamen sanatçıların elinde olacaktır.

Bir başka geliştirme ise ilkel şekillerin sayısının arttırılması ve genel model iyileştirmesi olabilir. Daha çok ilkel şekil, daha karmaşık ve sofistike şekillerin çıkmasına zemin hazırlayabilir. Halihazırda olan ilkel şekillerde ise algoritmik iyileştirmeler (optimizasyon vb.) yapılabilir. Şekil gramerinin yorumlanması sonucunda çıkan modelde ise vokselli görünümden kurtulmak için yumuşatma algoritmalarının kullanılma seçeneği sunulabilir.

Performans testi yapılırken de görüldüğü üzere yazılım hızlı çalışmaktadır. Bu da yazılımın bir oyun içerisinde gerçek zamanlı olarak kullanılabilmesine olanak sağlamaktadır. Yazılım model oluşturulurken kolaylıkla kullanılabilse de, bir oyun geliştirici oyununa direkt yazılımı

entegre etmek isterse zorlanma ihtimali olabilir. Yazılımın bütün kaynak kodu GitHub[3]'da bulunsa da, kodun başka bir yazılıma birleştirilmesi efor gerektiren bir iştir. Bunun için yazılımın bir paket halinde getirilip oyun geliştiricilerine entegrasyonda da erişilebilirlik sağlanmalıdır.

[3]GitHub linki: https://github.com/karabatut/Flying-Wing

KAYNAKÇA

[1] George Lucas (Director). *Star Wars: A New Hope*. 20th Century Fox, **1977**.

[2] Joss Whedon (Director). *Serenity*. Universal Pictures, **2005**.

[3] Christopher Nolan (Director). *Interstellar*. Warner Bros. Pictures, **2014**.

[4] James Strong (Director). *Doctor Who: The Impossible Planet*. BBC, **2006**.

[5] 'cyberpunk 2077' players are trying to make the 'perfect' v, sharing character creation presets. https://www.forbes.com/sites/paultassi/2021/02/05/cyberpunk-2077-players-are-trying-to-make-the-perfect-v-sharing?sh=3bcd0dd61f05, Erişim tarihi: **Eylül 2022**.

[6] FPires. Procedural spaceship generator. https://forum.unity.com/threads/procedural-spaceship-generator.445238/, Erişim tarihi: **Ekim 2021**.

[7] griffsnuff. Minecraft screenshot. https://www.deviantart.com/griffsnuff/art/Minecraft-screenshot-439905222/, Erişim tarihi: **Mart 2022**.

[8] Rogue (video game). https://en.wikipedia.org/wiki/Rogue_(video_game), Erişim tarihi: **Nisan 2022**.

[9] Perlin noise. https://en.wikipedia.org/wiki/Perlin_noise, Erişim tarihi: **Temmuz 2022**.

[10] Best minecraft 1.18 1.19 desert seeds for bedrock and java (september 2022). https://progameguides.com/minecraft/minecraft-desert-seeds/, Erişim tarihi: **Eylül 2022**.

[11] Michael Cook. Generate random cave levels using cellular automata. https://gamedevelopment.tutsplus.com/tutorials/generate-random-cave-levels-using-cellular-automata--gamedev-96 Erişim tarihi: **Mart 2022**.

[12] Jason Bevins. Example: Perlin worms. http://libnoise.sourceforge.net/examples/worms/, Erişim tarihi: **Mart 2022**.

[13] Grant Duncan. No man's sky: How i learned to love procedural art. https://www.youtube.com/watch?v=vcEA41eBOGs, Erişim tarihi: **Ocak 2022**.

[14] Crysis is 9 years old and still beautiful. https://steamcommunity.com/sharedfiles/filedetails/?id=633981387, Erişim tarihi: **Haziran 2022**.

[15] Alois Zingl. The beauty of bresenham's algorithm. http://members.chello.at/easyfilter/bresenham.html, Erişim tarihi: **Kasım 2021**.

[16] Bresenham's algorithm for 3-d line drawing. https://www.geeksforgeeks.org/bresenhams-algorithm-for-3-d-line-drawing/, Erişim tarihi: **Kasım 2021**.

[17] Z Crumley, Patrick Marais, and James Gain. Voxel-space shape grammars. **2012**.

[18] S. Dusterwald. *Procedural Generation of Voxel Worlds with Castles*. University of Waikato, **2015**.

[19] Markus Alexej Persson. Terrain generation, part 1. https://n0tch-blog-blog.tumblr.com/post/4231184692/terrain-generation-part-1, Erişim tarihi: **Mayıs 2022**.

[20] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. volume 25. **2006**. doi:10.1145/1141911.1141931.

[21] Rachel Hwang. Procedural building generation with grammars. https://www.youtube.com/watch?v=t-VUpX-xVo4, Erişim tarihi: **Mayıs 2021**.

[22] Doug Chiang. Doug chiang: The evolution of star wars design - designing episode i live panel at swcc 2019. https://www.youtube.com/watch?v=uYK1OXlm7Us, Erişim tarihi: **Temmuz 2022**.

[23] Kip Thorne. Interstellar – building a black hole – official warner bros. https://www.youtube.com/watch?v=MfGfZwQ_qaY, Erişim tarihi: **Haziran 2022**.

[24] Özüm İtez. Ankara apartmanları cumhuriyet'in gelişiminin Önemli bir fiziksel tanığı. https://www.arkitera.com/soylesi/ankara-apartmanlari-cumhuriyetin-gelisiminin-onemli-bir-fiziksel
Erişim tarihi: **Temmuz 2022**.

[25] Joseph Plotz. Typical parisian architecture in the 7th arrondissement. https://en.wikipedia.org/wiki/7th_arrondissement_of_Paris,
Erişim tarihi: **Temmuz 2022**.

[26] Christopher G. Jennings. Lindenmayer systems. https://cgjennings.ca/articles/l-systems/, Erişim tarihi: **Haziran 2022**.

[27] Jordan Sirani. The 10 best-selling video games of all time. https://www.ign.com/articles/best-selling-video-games-of-all-time-grand-theft-auto-minecraft
Erişim tarihi: **Ağustos 2022**.

[28] Phillip Koller. No man's sky review. https://www.polygon.com/2016/8/12/12461520/no-mans-sky-review-ps4-playstation-4-pc-windows-hello-games-son
Erişim tarihi: **Ocak 2022**.

[29] Rebecca Lavender. The zelda dungeon generator: Adopting generative grammars to create levels for action-adventure games. **2016**.

[30] Arttu Marttinen. Procedural generation of two-dimensional levels. **2017**.

[31] Jose Lazaro and Dirk Albrecht. From pixel to voxel in the clinical diagnosis. pages 1–5. **2016**. doi:10.1109/CONCAPAN.2016.7942370.

[32] Cryengine 2 shown at gdc 07–level editor demonstration 1/2. `https://www.youtube.com/watch?v=wDLTRXV1KHM`, Erişim tarihi: **Haziran 2022**.

Turnitin Orijinallik Raporu

İşleme kondu: 05-Eyl-2022 09:14 +03
NUMARA: 1892929851
Kelime Sayısı: 11312
Gönderildi: 1

OYUNLAR ICIN GRAMER TABANLI YORDAMSAL 3B MODEL GELISTIRME Batuhan Karaarslan tarafından

Benzerlik Endeksi	Kaynağa göre Benzerlik	
%7	Internet Sources:	%6
	Yayınlar:	%0
	Öğrenci Ödevleri:	%5