KULLANICI DİREKTİFLİ RASTGELE ARAZİ YÜZEYLERİNİN KOLAY VE HIZLI ÜRETİMİ

2011
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

Serpil EROĞLU

KULLANICI DİREKTİFLİ RASTGELE ARAZİ YÜZEYLERİNİN KOLAY VE HIZLI ÜRETİMİ

Serpil EROĞLU

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

KARABÜK

Haziran 2011

Serpil EROĞLU tarafından hazırlanan "KULLANICI DİREKTİFLİ RASTGELE ARAZİ YÜZEYLERİNİN KOLAY VE HIZLI ÜRETİMİ" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Baha ŞEN .....

Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 23/ 06/ 2011

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. Abdullah ÇAVUŞOĞLU (KBÜ)	.....
Üye : Doç. Dr. Fatih Vehbi ÇELEBİ (AÜ)	.....
Üye : Yrd. Doç. Dr. Baha ŞEN (KBÜ)	.....

.../.../2011

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Doç. Dr. Nizamettin KAHRAMAN .....

Fen Bilimleri Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Serpil EROĞLU

ÖZET

Yüksek Lisans Tezi

KULLANICI DİREKTİFLİ RASTGELE ARAZİ YÜZEYLERİNİN KOLAY VE HIZLI ÜRETİMİ

Serpil EROĞLU

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Yrd. Doç. Dr. Baha ŞEN
Haziran 2011, 74 sayfa

Bilgisayar sistemlerindeki gelişmelere paralel olarak grafik kartlarının işlem kapasiteleri oldukça artmıştır. İşlem kapasitesinin artması sonucu yüksek çözünürlüklü 3 boyutlu uygulamalar yaygınlaşmaktadır. 3B uygulamalar oyun ve film gibi eğlence sektöründe kullanıldığı gibi çeşitli uçuş, tank, sürüş vb. eğitim simülatörlerinde de kullanılmaktadır. Bu tez çalışmasında sanal gerçeklik ortamlarında kullanılmak üzere yükseklik haritalarının oluşturulması ve 3B modeli gerçekleştirilmiştir. Yükseklik haritalarının oluşturulmasında rastgele yöntemlerden olan Fault, Çember, Parçacık Ekleme, Dörtgensel Prizma Ekleme ve FFT algoritmaları kullanılmıştır. Oluşturulan yükseklik haritaları 3B modellenebildiği gibi gerçek dünyaya ait verilerin de modeli oluşturulabilmektedir. Bunun için çeşitli harita dosyaları kullanılmaktadır. DEM, DTED, SRTM bunlara verilebilecek örnekler arasındadır. Çalışmanın ikinci adımı kullanıcının isteğine göre arazi modeli üzerinde değişiklik yapılmasıdır. Bunun için, kullanıcı istediği herhangi bir noktanın

yükseklik değerini kullanıcı ara yüzü yardımıyla gerçek zamanlı değiştirebilmektedir. Arazi yüzeyi üzerinde yapılan düzenleme işlemleri ekranda gerçek zamanlı olarak kullanıcıya sunulmaktadır. Bu işlemlerdeki CPU hesaplama performansını artırmak için NVIDIA CUDA teknolojisi kullanılmaktadır. CUDA sayesinde iş yükü ekran kartı üzerinde bulunan işlemcilere paralel olarak dağıtılmaktadır. Kullanılan bu paralel programlama tekniği sayesinde performans artışı elde edilmektedir. Gerçekleştirilen uygulama JAVA ve OpenGL tabanlıdır.

Anahtar Sözcükler : GIS, yükseklik haritası üretimi ve modifikasyonu, DEM, DTED, CUDA, gerçek zamanlı uygulama, rastgele yükseklik haritası.

Bilim Kodu : 902. 1.014

ABSTRACT

M.Sc. Thesis

FAST AND EASY GENERATION OF USER DIRECTED RANDOM TERRAINS

Serpil EROĞLU

Karabük University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Baha ŞEN
June 2011, 74 pages

Capacity of graphic cards has been increased in parallel with developments on computer systems. As a result of increasing processing capacity high-resolution 3D applications have been become popular. 3D applications are used for entertainment industry like game and film and also used for different training simulators such as flight, driving, tank etc. In this study, establishment of height map to be used in virtual reality environments and 3D model have been realized. Fault Algorithm, Circle Algorithm, Particle Decomposition Algorithm, Rectangular Prism Algorithm and FFT algorithm have been used for producing random height maps. As the created height map can be modeled 3D, the models of real-world data can also be created. The various map files such as DEM, DTED, and SRTM are used for this process. Second step of presented study is modification of terrain model according to users' demands. User can modify height value of the any point on the model with the help of user interface at real time all modifications on the terrain surface have been

presented to the user on screen at real time. NVIDIA CUDA technology is used to improve performance of applications. Thanks to CUDA technology, work load is delivered parallel to processors which are on the graphic card. With this parallel programming method, performance increment has been obtained. Realized application is based on JAVA programming language and OpenGL graphic library.

Key Words : Geographical information system, generation and modification of height map, DEM, DTED, CUDA, real time application, random height map.

Science Code : 902. 1.014

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Yrd. Doç. Dr. Baha ŞEN' e sonsuz teşekkürlerimi sunarım.

Sevgili aileme manevi hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

KISALTMALAR

SYM	:	Sayısal Yükseklik Modeli
SAM	:	Sayısal Arazi Modeli
DSM	:	Digital Surface Model
DGM	:	Digital Ground Model
DTM	:	Digital Terrain Model
CBS	:	Coğrafi Bilgi Sistemleri
CUDA	:	Compute Unified Device Architecture
GIS	:	Geographic Information System
USGS	:	United States Geological Survey
SRTM	:	Shuttle Radar Topography Mission
DTED	:	Digital Terrain Elevation Data
DEM	:	Digital Elevation Model
UHL	:	User Header Label
DSI	:	Data Set Identification Record
ACC	:	Accuracy Description Record
GPU	:	Graphics Processing Unit
CPU	:	Central Processing Unit
GPGPU	:	General-purpose computation on Graphics Processing Units
NIMA	:	National Imagery and Mapping Agency
2B	:	2 Boyut
3B	:	3 Boyut
2D	:	2 Dimension
3D	:	3 Dimension

BÖLÜM 1

GİRİŞ

Bilgisayar sistemlerindeki gelişmelere paralel olarak grafik kartlarının işlem kapasiteleri oldukça artmıştır. İşlem kapasitesinin artması ile yüksek çözünürlüklü 3 boyutlu uygulamalar yaygınlaşmaktadır. 3B uygulamalar oyun ve film gibi eğlence sektöründe kullanıldığı gibi çeşitli uçuş, tank, sürüş vb. eğitim simülatörlerinde de kullanılmaktadır. 3B uygulamaların en büyük sorunlarından biri de sanal gerçeklik ortamı yaratmak için kullanılan arazi modellerinin oluşturulma teknikleridir. Bu amaçla çeşitli çalışmalar gerçekleştirilmiştir.

İsteğe uygun arazi modeli üretmek amacıyla minimum seviyede kontrol edilebilir bir teknik sunulmuştur. Bu teknik sürekli rastgele çokgenler oluşturarak araziyi üretmektedir [1].

Büyüyen bir şehirde kentsel arazi kullanımını belirlemek için ajan tabanlı bir yaklaşım kullanılmıştır. Böylece coğrafyacıları ve teknik bilgisi olmayan normal kullanıcıları tek bir noktada toparlayan bir araç(program) sunuldu. Üretilen çözümler kontrol edilebilirlik özelliği taşımakla birlikte tasarımcıların parametreleri sezgilerine göre değil de jeomorfolojik açıdan gerçekçi verilere göre vermeleri gerekmektedir [2].

Makine öğrenmesi kullanılarak arazi üretimi çalışmaları yapılmıştır. Bu çalışmalarda makineye örnek yükseklik haritaları sunularak makinenin bu harita verilerini modellemesi öğretilmiştir. Daha sonra bu modelden yeni araziler üretilmeye çalışılmıştır [3].

Yapay arazi modeli üretimi için yeni bir teknik geliştirilmiştir. Bu teknik var olan haritalardan yapay harita elde etmeye dayanmaktadır. Mevcut

DEM içerisindeki yüksek çözünürlüklü arazi verilerini ayıklayarak, düşük çözünürlüklü arazi üretmeyi amaçlamışlardır [4].

GIS haritaları kullanarak bu haritalar üzerinde belirlenen bazı kurallar çerçevesinde değişken şehir ortamlarını oluşturan sanal şehirlerin simülasyonunu üzerine yapılan çalışmalar mevcuttur [5].

Yazılım ajanı kavramını kullanılarak kontrol edilebilen bir sistem geliştirmiştir. Geliştirilen bu sistemde kullanıcılar çeşitli parametreler aracılığıyla istedikleri özellikte sanal araziler oluşturabilmektedirler. Geliştirilen yazılımda arzu edilen arazi modeline göre seçenekler bulunmaktadır. Sahil kenarı, ormanlık arazi, kayalık arazi oluşturma gibi seçenekleri bulunmaktadır [6].

Hazırlanan bu tez çalışmasının ilk bölümünde literatür taraması ve çalışmanın kısa özeti verilmiştir. İkinci bölümde, Sayısal yükseklik modeli hakkında genel bir bilgi ve üretilme teknikleri verilmiştir. Üçüncü bölümünde sayısal yükseklik modeli oluşturmak için kullanılabilecek rastgele algoritmalar anlatılmıştır. Dördüncü bölümde, performansı artırmak için kullanılan paralel programlama ve teknikleri hakkında bilgi verilmiş olup beşinci bölümde ise uygulamanın geliştirilmesi süresince kullanılan yöntemler detaylı olarak anlatılmıştır. Son bölüm olan altıncı bölümde bu çalışma sayesinde elde edilen sonuçlar sunulmaktadır.

BÖLÜM 2

SAYISAL YÜKSEKLİK MODELİ

Yeryüzü 3B düzensiz bir yüzeydir. Bu yüzeyi matematiksel olarak tanımlamak oldukça zordur. Bunun için sonsuz sayıda noktaya ihtiyaç duyulur. Bunun yerine belirli aralıklarda nokta kümeleri seçilir ve yüzey matematiksel olarak temsil edilmeye çalışılır. Yeryüzünün bu biçimde temsili Sayısal Arazi Modeli (SAM) olarak tanımlanır ve yüzeyin bu biçimde temsili yerbilimlerinde, çok sayıda mühendislik alanında, askeri uygulamalarda ve diğer birçok alanda yaygın olarak kullanılır. SAM'ın yanı sıra Sayısal Yükseklik Modeli (SYM), Digital Ground Model(DGM), Digital Terrain Model (DEM) ve Digital Surface Model (DSM) terimleri de kullanılmıştır [7,8].

Sayısal arazi modeli kavramı ilk olarak, 1958 yılında Miller ve Laflamme tarafından yol projelerinin sayısallaştırılması amacıyla ortaya atılmıştır. Başlangıçta uygulamalar, eş yükseklik eğrilerinin oluşturulması ile sınırlı kalmıştır. Genel olarak grafik işlemlerden bilgisayar işlemlerine geçilmesi ile SAM veya SYM tek başına bir ürün olmuştur [9,10].

Literatürde Sayısal Arazi Modelinin birçok tanımı bulunmaktadır. Sayısal arazi modeli gerçekte, arazinin topografik yüzeyinin, matematiksel tanımı yapılabilen bir yüzeyle temsil edilmesi problemidir. Bu problem, topografik yüzey üzerinde üç boyutlu koordinatlarıyla tanımlanmış dayanak noktalarına bağlı olarak çözülebilmektedir [11].

Bazı çalışmalarda sayısal arazi modeli ve sayısal yükseklik modeli arasında kesin bir ayrım yapılmamıştır. Ancak SAM'lar Sayısal Yükseklik Modellerinden (SYM) farklı olarak yeryüzü üzerindeki detayları yansıtmazlar. Sadece arazi üzerindeki belirli noktaların yükseklik verisi tutulur. Fakat bir SYM, arazi üzerindeki bina, bitki

örtüsü, orman v.b. farklı yükseklik değerlerine sahip detayları içerirken yani görünür yeryüzünü yansıtırken SAM' lar bu detayları içermez. Tüm bu özellikleriyle SAM' lar yeryüzü topografyasını en basit şekilde yansıtan en genel ve yaygın model olarak tanımlanmaktadırlar [12].

2.1. SAYISAL YÜKSEKLİK HARİTALARI ELDE EDİLME YÖNTEMLERİ

Sayısal yükseklik modelinin oluşturulmasında en önemli aşama verilerin toplanmasıdır. Günümüzde çok çeşitli veri toplama yöntemleri bulunmaktadır. Teknolojinin gelişmesiyle birlikte veri toplama teknikleri de gelişim göstermiştir. Başlangıçta basit yöntemlerle elde edilen sayısal yükseklik modelleri günümüzde yerini çok yüksek çözünürlüklü haritalar elde edebilen uydulara bırakmıştır. Genel olarak bakıldığında veri toplama tekniklikleri üçe ayrılmaktadır.

2.1.1. Yersel Tekniklerle SYM Üretimi

Yersel tekniklerle SYM üretimi için çeşitli yersel ölçme aletleri kullanılmaktadır. Bu aletler aracılığıyla SYM'si oluşturulmak istenen araziden nokta nokta konum ve yükseklik verisi toplanır. Bunun için yersel lazer tarayıcı (LYS) kullanılmaktadır. LYS'ler 3B yüksek çözünürlüklü verileri yüksek doğrulukta ve kısa zamanda belirlememizi sağlar. Piyasada farklı özelliklerde LYS'ler bulunmaktadır.

Lazer tarayıcılar yatay ve düşey eksende çeşitli açısal aralıklarla uzaklık ölçerler. Ölçtükleri tüm noktaların bir araya gelmesine "nokta bulutu" denilmektedir. Nokta bulutu bir görüntü oluşturmuş olur. Tarayıcı bu nokta bulutunu dönen bir kafa ile lazer ışını göndererek elde eder [13].

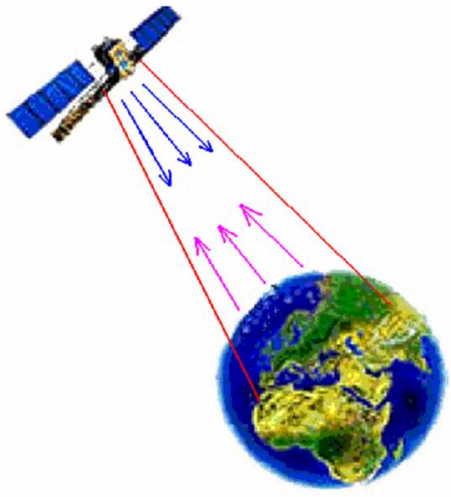
2.1.2. Fotogrametri Tekniğiyle SYM Üretme

Fotogrametri tekniğinde amaç yeryüzünün fotograflarının çekilmesi ile modelinin oluşturulabilmesidir. Bu teknikte fotoğrafların çekim yeri önemlidir. Çekim yerine göre, yersel fotogrametri ve hava fotogrametrisi olmak üzere ikiye ayrılmaktadır. Yersel fotogrametride resimler yerden çekilirken hava fotogrametrisinde uçuş

sırasında çekilen hava fotoğraflarından bir noktaya ait yükseklik verisi tespit edilmeye çalışılır [14].

2.1.3. Uzaktan Algılama Tekniğiyle SYM Üretimi

Uzaktan algılama yeryüzene herhangi bir temas olmaksızın nesnelerin gözlenmesi ve ölçülmesini sağlar. Gözleri görmeyen yarasaların yönlerini bulabilmeler ve nesnelere çarpmadan uçabilmeleri için ses dalgası gönderip bu dalganın geri dönüşüne göre hareket etmeleri uzaktan algılamanın temelini oluşturmaktadır. Uzaktan algılama verileri kameralar, sensörler ile donatılmış uçaklar ve uydular tarafından sağlanmaktadır. RADAR (Radio Detectig and Ranging) uzaktan algılama amaçlı birçok aracın kullandığı bir sistemdir. RADAR Şekil 2.1'de gösterildiği gibi kendi kaynağından hedefe enerji gönderir ve hedeften geri yansıyan enerjiyi alır ve kaydeder. Yağmurlu, sisli gibi hava koşulları ve mevsim fark etmeksizin her koşulda ve her zaman algılama yapılabilmektedir. Çizelge 2.1 günümüzde radar ölçme tekniğiyle çalışan birçok uzay aracı yeryüzünün bilgisini toplamıştır.



Şekil 2.1. RADAR'ın çalışma metodu [15].

Çizelge 2.1. Radar tekniği kullanan uzay araçları [16].

Uzay Aracı	**Ülke**	**Fırlatılış Tarihi**	**Yörüne Yüksekliği**
SRTM	Amerika,Almanya, İtalya	11 Şubat 2000	233 km
JERS-1	Japonya	11 Şubat 1992	568 km
RADARSAT-1	Kanada	4 Kasım 1995	798 km
ENVISAT	ESA'nın 13 üyesi Kanada,Fransa, İngiltere	1 Mart 2002	800 km
ALOS	Japonya	24 Ocak 2006	692 km

Uydulardan elde edilen görüntülerin uçaklarla elde edilen görüntülere oranla maliyetleri daha düşüktür. Elde edilen görüntüler sayısal formatta olduğu için doğrudan bilgisayar ortamında işlenebilecek formattadır.

Ülkemiz de 27 Eylül 2003 tarihinde BilSAT 1 isimli ilk uzaktan algılama uydusunu fırlatmıştır. BilSAT 1 uydusundan 12,6 m çözünürlükte siyah-beyaz (pankromatik) ve 27,6 m çözünürlüklü renkli (multispektral) uydu görüntüleri elde edilebilmektedir [17].

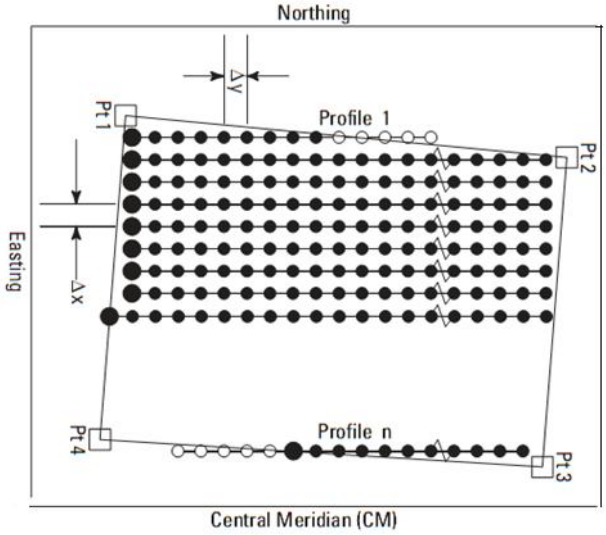
2.2. GIS HARİTALARI

GIS "Geographic Information System" olarak ifade edilen terim Türkçede "Coğrafi Bilgi sistemi" olarak ifade edilmektedir. Özetle Coğrafi Bilgi Sistemleri (CBS, GIS) konuma dayalı verilerle, sözel verilerin toplanması, saklanması ve analizi işlemlerini gerçekleştirilen bilgi sistemleridir [18].

GIS konuma dayalı verileri kullanıcılara farklı standartlarda ve farklı dosya formatlarında sunmaktadır. Bu standartları geliştiren çeşitli kurumlar bulunmaktadır. Her bir standardın ayrı kuralları bulunmaktadır. İçlerinde tuttukları bilgi formatları da farklıdır. Bu çalışmada USGS (United States Geological Survey) Amerikan Yerbilimsel Araştırma Kurumu'nun hazırlamış olduğu harita formatları incelenmiştir.

2.2.1. DEM Haritaları

Amerika Birleşik Devletleri Jeolojik Araştırmalar (USGS, United States Geological Survey) Kurumu tarafından oluşturulan açık standartlı bir harita yapısıdır. USGS resmi sitesinden ücretsiz olarak indirilebilmektedir. Kayıtlar ASCII kodlanmış bir biçimde dosyada tutulmaktadır. Veriler Şekil 2.2'de gösterildiği formatta tutulmaktadır.



Şekil 2.2. DEM haritası dosya yapısı [19].

Bu yapıda Δx ve Δy noktalar arasında bulunan uzaklığı ifade etmektedir. Pt tutulan noktaların köşelerini göstermektedir. Verilerin bu şekilde tutulmasına grid yapısı adı verilmektedir.

2.2.2. SRTM Haritaları

Space Radar Topography Mission (SRTM) Amerikan NASA kurumu tarafından yaklaşık 60° kuzey ve güney enlemleri arasında kalan tüm kara parçalarının sürekli ve yüksek çözünürlüklü sayısal yükseklik modelini elde etmek amacıyla gerçekleştirilmiş bir projedir [20,21].

Bu amaçla geliştirilen uzay mekiği 2000 yılı Şubat ayında fırlatılmış yapay açıklıklı radar (SAR) yöntemi ile 11 gün boyunca veri toplamıştır. Bu yöntemde yeryüzüne mikrodalga sinyaller gönderilerek güneşin konumundan, hava koşullarından etkilenmeden veri toplamak mümkün olmaktadır

SRTM verileri üç farklı çözünürlükte hazırlanmıştır. Orijinal veriler coğrafi koordinatlarda 1", 3" ve 30"çözünürlükte grid biçiminde sunulmaktadır. Coğrafi koordinatlar derece biriminde WGS84, yükseklikler metre biriminde EGM96 jeoidine göre tanımlanmıştır.

SRTM formatında başlangıç noktasının enlem ve boylamı dosya isminde (örneğin N34E032.hgt) belirtilerek, dosyada satırlar halinde 2 byte uzunluğunda tamsayı olarak yükseklikler yer almaktadır. Srtm kayıtları 4 ana gruba ayrılmıştır.

- UHL (User Header Label)
- DSI (Data Set Identification Record)
- ACC (Accuracy Description Record)
- DATA (Data Record)

UHL grubunda dosya ile ilgili genel bilgiler yer alır. Başlangıç boylamı, enlemi, Toplam enlem sayısı, boylam sayısı bilgileri gibi temel veriler bulunur. UHL kaydının yapısı Çizelge 2.2.'de gösterilmiştir. UHL kaydında bulunan her bir byte verisinin dosya açısından ne ifade ettiği bu çizelge yardımıyla görülmektedir.

Çizelge 2.2. SRTM dosyası UHL kayıt yapısı [22].

Alan	**Byte**	**İçerik**	**Açıklama**
1	1-3	UHL	Kayıt başlığı
2	4	1	Standart
3	5-12	DDDMMSSH	Başlangıç boylamı
4	13-20	DDDMMSSH	Başlangıç enlemi
5	21-24	SSSS	Boylam aralığı
6	25-28	SSSS	Enlem aralığı
7	29-32	Boş	
8	33-35	U	Güvenlik kodu
9	36-47	Referans Numarası	
10	48-51	Boylam Sayısı	Toplam boylam sayısı
11	52-55	Enlem Sayısı	Toplam enlem sayısı

12	56	Çoklu Doğruluk	0-Tek 1-Çok

DSI kaydı Çizelge 2.3'de görüldüğü gibi SRTM doyası ile ilgili kimlik ve güvenlik bilgileri tutar. Kayıt 648 ASCII karakterden oluşur ve uzunluğu sabittir. Bu kayıtta dosyanın üretici kodu, güvenlik kodu gibi bilgiler yer alır.

Çizelge 2.3. SRTM dosyası DSI kayıt yapısı [22].

Alan	Byte	İçerik	Açıklama
1	1-3	DSI	Kayıt adı
2	4	U	Güvenlik kodu
3	5-6	boşluk	
4	7-33	Boşluk	
5	34-59	Boşluk	
6	60-64	Dted1 yada	
7	65-79	00000000	Referans numarası
8	80-87	Boşluk	
9	88-89	01-99	Veri edisyon numarası
10	90	A-Z	Versiyonu
11	91-94	0000	0
12	95-98	0000	0
13	99-102	0000	0
14	103-110	0000	0
15	111-126	Boşluk	
16	127-135	Boşluk	
17	136-137	00	
18	138-141	0000	Üretim tarihi (YYMM)
19	142-144	W87	Düşey datum
20	143-149	W6s84	Yatay datum
21	130-159	GeMoS2.0.0	Sayısallaştırma sistemi
22	160-163	YYMM	Derleme zamanı
23	164-185	Boşluk	
24	186-194	DDMMSS.SH	Enlem bilgileri
25	195-204	DDMMSS.SH	Boylam bilgileri
26	205-211	DDMMSSH	Haritadaki enlem güney batı köşesi
27	212-219	DDMMSSH	Haritadaki boylamın güney batı köşesi
28	220-226	DDMMSSH	Haritadaki enlem kuzey batı köşesi
29	227-234	DDMMSSH	Haritadaki boylamın kuzey batı köşesi
30	235-241	DDMMSSH	Haritadaki enlem kuzey doğu köşesi
31	242-249	DDMMSSH	Haritadaki boylamın kuzey doğu köşesi
32	250-256	DDMMSSH	Haritadaki enlem güney doğu köşesi
33	257-264	DDMMSSH	Haritadaki boylamın güney doğu köşesi
34	265-273	00000	
35	274-277	SSSS	
36	278-281	SSSS	
37	282-285	0000-9999	Enlem satırı sayısı
38	286-289	0000-9999	Boylam satırı sayısı
39	290-291	00-99	Hücre göstergesi
40	292-392	00-99999999	Yüzdelik gösterge
41	393-492	Geoid kıvrımı	
42	498-648	boş	boş

ACC kaydı SRTM ile ilgili doğruluk bilgilerini tutar. Kayıt 2700 ASCII karakterden oluşur. Çizelge 2.4'de ACC kayıt yapısı görülmektedir. Bazı kısımları gelecekte doldurulmak üzere boş bırakılmıştır.

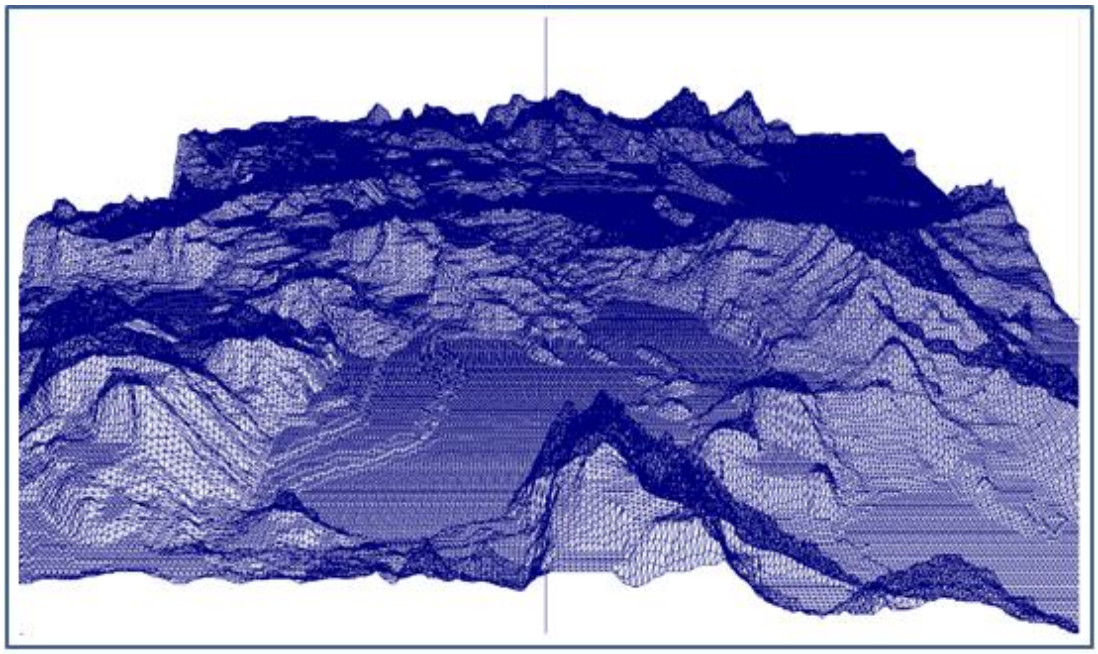
Çizelge 2.4. SRTM dosyası ACC kayıt yapısı [22].

Alan	**Byte**	**İçerik**	**Açıklama**
1	1-3	ACC	
2	4-7	NA$$	Ulaşılamıyor
3	8-11	NA$$	Ulaşılamıyor
4	12-15	NA$$	Ulaşılamıyor
5	16-19	NA$$	Ulaşılamıyor
6	20-23	Boşluk	Gelecekte kullanılmak üzere ayrılmış alan
7	24	Boşluk	Gelecekte kullanılmak üzere ayrılmış alan
8	25-55	Boşluk	Gelecekte kullanılmak üzere ayrılmış alan
9	56-57	00	Gelecekte kullanılmak üzere ayrılmış alan
10	58-2613	Boşluk	Gelecekte kullanılmak üzere ayrılmış alan
11	2614-2631	Boşluk	Gelecekte kullanılmak üzere ayrılmış alan
12	2632-2700	Boşluk	Gelecekte kullanılmak üzere ayrılmış alan

DATA kaydında ise yükseklik verileri tutulur. Çizelge 2.5'de görüldüğü gibi bu kayıt yapısında haritada bulunan noktaların yükseklik verisi tutulur.

Çizelge 2.5. SRTM dosyası DATA kayıt yapısı [22].

Alan	**Byte**	**İçerik**	**Açıklama**
1	1	170	Kayıt adı
2	3	blok sayısı	İlk blok sıfır numaradan başlar
3	2	Enlem Sayısı	Meridyen sayısını verir
4	2	Boylam Sayısı	Boylam sayısını verir.
5	2	Yükseklik	Yükseklik verisi metre olarak verilir.
6	4	Toplam	Blok verilerinin toplamı tutulur.



Şekil 2.3. Susanville, (Kalifornia) 3B modeli.

SRTM dosyalarında bulunan bu kayıt yapıları ile ihtiyaç duyulan bilgiye erişim kolaylığı sunulmaktadır. Kayıt bloklarının uzunlukları sabit olduğundan hangi bilgi isteniliyorsa o kayıt bloğuna gidilerek bilgiler okunabilmektedir. Şekil 2.3'de Susanville (California) şehrinin SRTM dosyası okunarak 3B modeli sunulmaktadır.

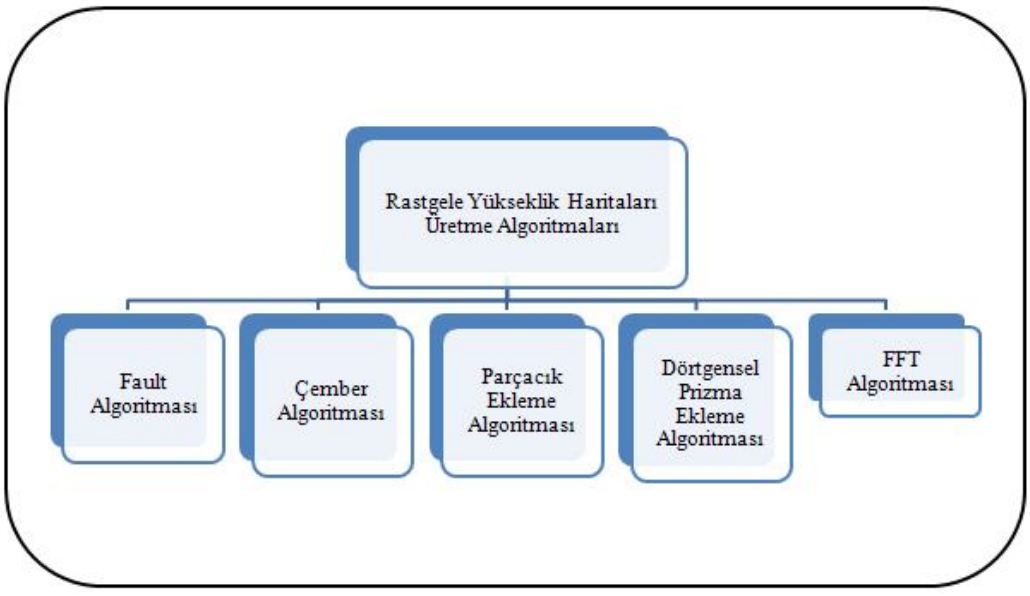
2.2.3 DTED Haritaları

Askeri uygulamalarda konumsal bilgiler ve bu bilgilerin doğruluğu çok önemlidir. Bu amaçla Ulusal Görüntü Haritalama Ajansı (NIMA, USA) DTED adı verilen bir arazi yüzeyinin yükseklik bilgilerini grid yapısını biçiminde tutan bir standart geliştirdi. DTED arazi profili çıkarma, 3 boyutlu arazi sanallaştırma, uçuş modelleme ve simülasyon gibi bir çok uygulamalarda kullanılan harita dosyasıdır. Bu harita yapısında deniz ve göllerin yükseklik değerleri sıfır olarak gösterilir. Arazi yükseklik birimi metredir [22].

BÖLÜM 3

RASTGELE SAYISAL YÜKSEKLİK HARİTALARI ÜRETME ALGORİTMALARI

Doğada çok çeşitli arazi yüzeyleri bulunmaktadır. Bunlar dağlık araziler, çöller, sahiller, vb olmak üzere çeşitlendirilebilir. Arazi yüzeyleri kullanılacak uygulamaya göre değişiklik gösterebilir. Örneğin bir uçuş simülatöründe kullanılacak olan arazi yüzeyi ile bir bilgisayar oyununda kullanılacak olan arazi çeşidi birbirinden farklıdır. Uçuş simülatöründeki amaç pilot adayının dağlık alanlarda keskin dönüşler yapmasını sağlamak ve manevra kabiliyetini ölçmektir. Arazi modelini bu amaca hizmet edecek şekilde tasarlamak gerekir. Bir bilgisayar oyununda ise istenilen arazi yapısı tamamen farklıdır. Oyunun yapısına göre orman, sahil, dağlık alanlar olabildiği gibi daha düzlemsel arazi modelleri de tercih edilir. Bu ihtiyacı gidermek için arazideki yükseklik verilerinin düzlemsel arazi modeline uygun bir biçimde dönüştürülmesi gerekir. Bu amaçla çeşitli rastgele sayısal yükseklik haritası oluşturma algoritmaları mevcuttur.



Şekil 3.1. Rastgele SYM üretme algoritmaları.

Şekil 3.1. rastgele sayısal yükseklik üretme algoritmalarını göstermektedir. Rastgele algoritmaların üretilmesinde çok çeşitli kontroller bulunmaktadır. Her algoritmanın yapısı birbirine benzemekle birlikte üretilen arazi modelleri farklılık göstermektedir.

3.1. FAULT ALGORİTMASI

Algoritma başlangıç değerleri verilmiş veya sıfırlanmış bir yükseklik haritası kullanır. Yükseklik haritası nxn boyutlarında yükseklik verilerinin tutulduğu bir grid yapısıdır. Algoritmanın mantığı yükseklik haritası üzerinde rastgele bir çizgi çizilmesine dayanır. Bu çizgi yükseklik haritasını iki parçaya bölmektedir. Parçalardan birinin yükseklik verisi artırılırken diğer parçanın yükseklik verisi azaltılır. Böylece yükseklik haritası üzerinde bir fay hattı oluşur. Bu şekilde iterasyon sayısı artırıldığında daha gerçekçi bir arazi modeli oluşmuş olur [23]. Bin iterasyon sonucunda oluşan arazi yapısı Şekil 3.2'de gösterildiği gibidir. Algoritmadaki problem, arazi üzerinde seçilen rastgele çizginin araziyi iki parçaya ayırıp ayırmadığını bulmaktır. Yani öyle bir çizgi seçilmeli ki bu çizgi araziyi iki parçaya ayırmalıdır. Bunun için düzlem üzerinde bulunan doğru denkleminden faydalanılır. Düzlem üzerinde bulunan bir doğrunun denklemi Eşitlik 3.1'de gösterilmiştir

$$a * x + b * z - c = 0 \tag{3.1}$$

Bu formülde a, b ve c değerleri çizginin bulunması için önemlidir.

w: Arazinin genişliği (3.2)
l=Arazinin uzunluğu
v = rand(); //Rastgele bir sayı
a = sin(v);
b = cos(v);

$$d = \sqrt{\left(w^2 + l^2\right)}$$

$$c = rand() * d - {}^{d}\!/_{2}$$

a,b,c değerleri Eşitlik 3.2 sayesinde bulunmuş olacaktır. Bu değerler bulunduktan sonra araziyi iki parçaya bölen doğru elde edilmiştir. Böylece arazi üzerindeki her bir noktanın bu doğruya göre durumu tespit edilir.

$P(x_i, z_i)$ noktasının bir doğruya olan uzaklığını bulmak için Eşitlik 3.3'den faydalanılmaktadır.;

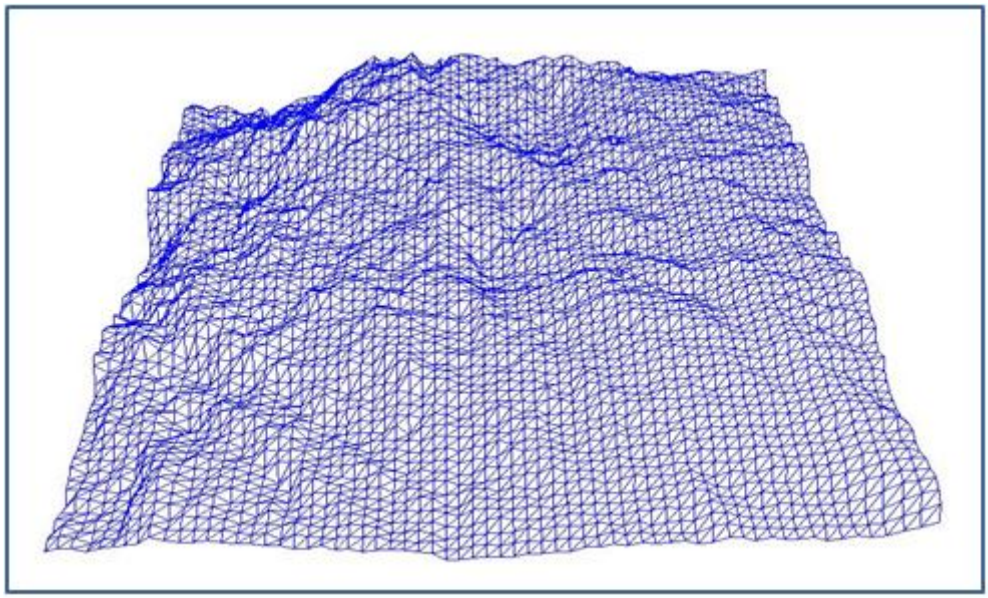
$$r = a * x_i + b * z_i - c \tag{3.3}$$

Elde ettiğimiz r değerine göre 3 durum söz konusudur.

- *r=0* için nokta doğru üzerinde bulunmaktadır.
- *r<0* için doğrunun düzlemi kestiği bir bölgede
- *r>0* için doğrunun düzlemi kestiği diğer bölgede

olduğu anlamına gelir. Noktanın doğrunun hangi bölgesinde olduğu önemli değildir. r durumuna göre o noktanın yükseklik değeri artırılır veya azaltılır. Aşağıda bu işlemin sembolik kodu verilmiştir.

Arazi üzerindeki her bir (t_x,t_z) için
{
*Eğer((a_x*t_x+b*t_z- c) > 0) ise*
(t_x,t_z) yüksekliğini artır
Değil ise
(t_x,t_z) yüksekliğini azalt
}



Şekil 3.2. Fault algoritması ile oluşturulmuş arazi modeli.

3.2. ÇEMBER ALGORİTMASI

Çember Algoritması, Fault Algoritmasına benzer bir yapı taşımaktadır. Fault algoritmasından farkı, yükseklikleri değişecek olan noktaların belirlendiği kısımdır. Fault algoritmasında harita üzerinde rastgele bir çizgi çizilirken çember algoritmasında harita üzerinde belirli bir yarıçapı olan bir çember seçilir. Çember içerisinde kalan noktaların yükseklikleri artırılır veya azaltılır çember dışında kalan noktaların yükseklikleri sabit kalır.

Bu algoritmayı uygulamak için öncelikle arazi içerisinde rastgele bir nokta seçilir. Bu nokta çemberin merkezi olmaktadır. Merkezi a,b yarıçapı r olan çemberin denklemi aşağıdaki Eşitlik 3.4'de gösterilmiştir.

$$r^2 = (x-a)+(z-b) \tag{3.4}$$

Harita üzerindeki her bir noktanın yarıçapı belli olan bu çembere olan uzaklığını bulmak için Eşitlik 3.5 kullanılır.

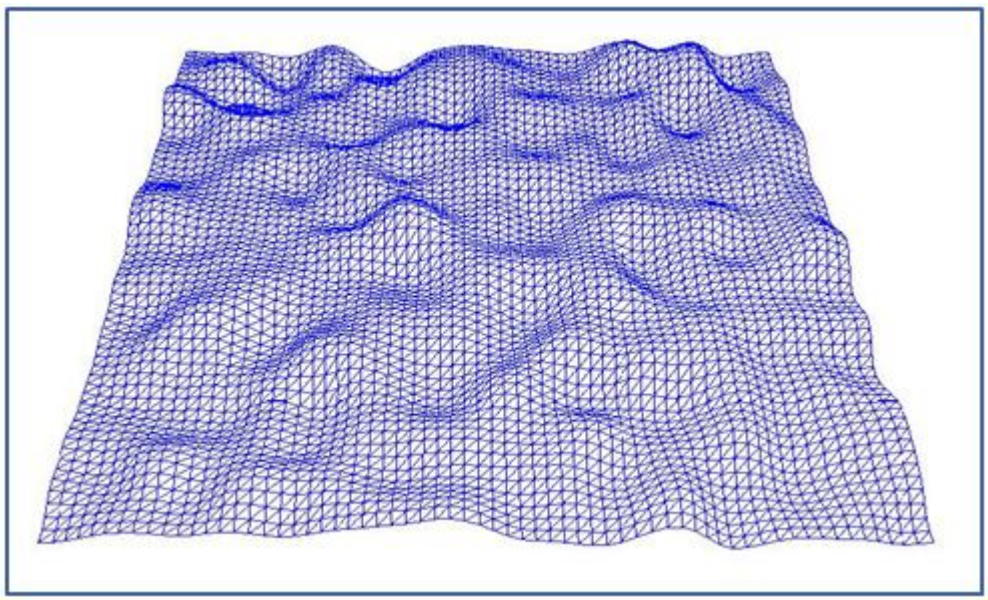
$$r = \sqrt{(\mathrm{x_i} - X)^2 + (\mathrm{z_i} - Z)^2} \tag{3.5}$$

- P(X,Z) çemberin merkezi
- P(x_i,z_i) Koordinat düzleminde herhangi bir nokta
- r: Bir noktanın çembere olan uzaklığı

Elde ettiğimiz r değerine göre 3 durum söz konusudur.

- *r=0* için nokta çember üzerinde
- *r<0* için nokta çember içerisinde
- *r>0* için nokta çember dışında bulunmaktadır.

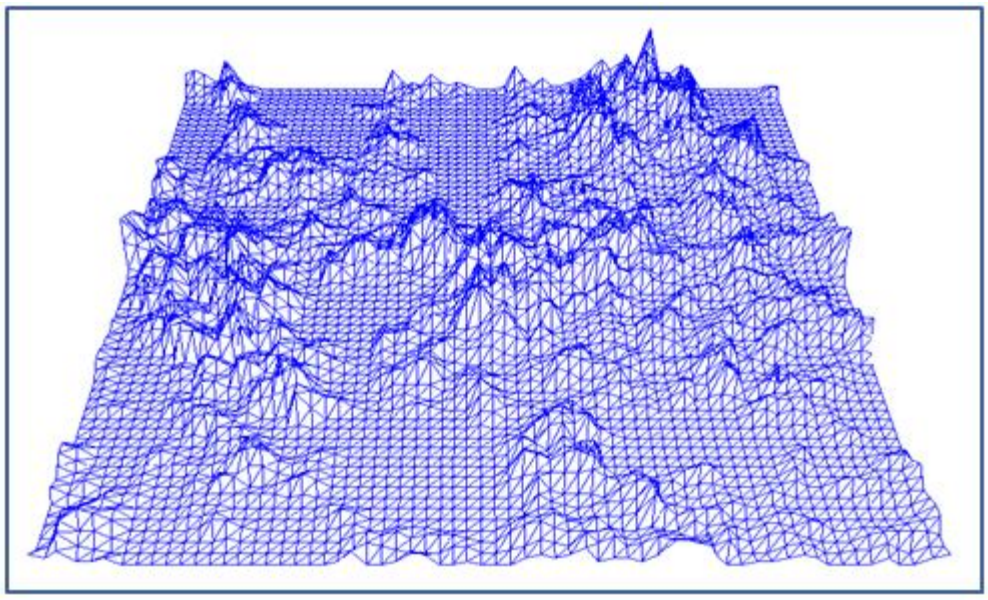
İterasyon sayısının artması sonucunda Şekil 3.3'de gösterildiği gibi arazi üzerinde dalgalanmalar meydana gelecektir. Portakal kabuğunu andıran bir arazi modeli elde edilmiş olur. Çember yarıçapının küçük bir değer seçilmesi ile daha farklı bir model elde edilebilir.



Şekil 3.3. Çember algoritması ile oluşturulmuş arazi modeli.

3.3. PARÇACIK EKLEME ALGORİTMASI

Bu algoritma arazi üzerine rastgele parçacıkların eklenmesiyle oluşmaktadır. Arazi üzerinde herhangi bir nokta seçilir ve o noktanın değeri istenilen miktar kadar artırılır. Programcının isteğine göre seçilen bu noktanın yanındaki herhangi bir noktanın da yükseklik değeri artırılır. Bu algoritmada herhangi bir kısıtlama bulunmamakla birlikte oluşturulan parçacıklar tamamıyla programcının hayal gücüne bırakılmıştır. Şekil 3.4'de bu algoritma sonucu oluşmuş bir arazi örneği görülmektedir.

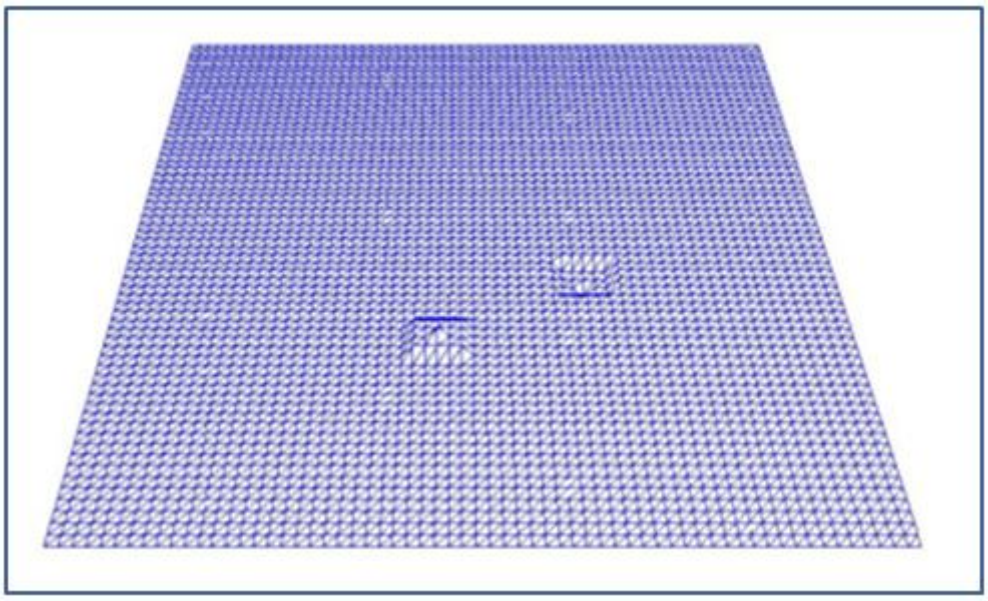


Şekil 3.4. Parçacık ekleme algoritmasıyla oluşmuş arazi modeli.

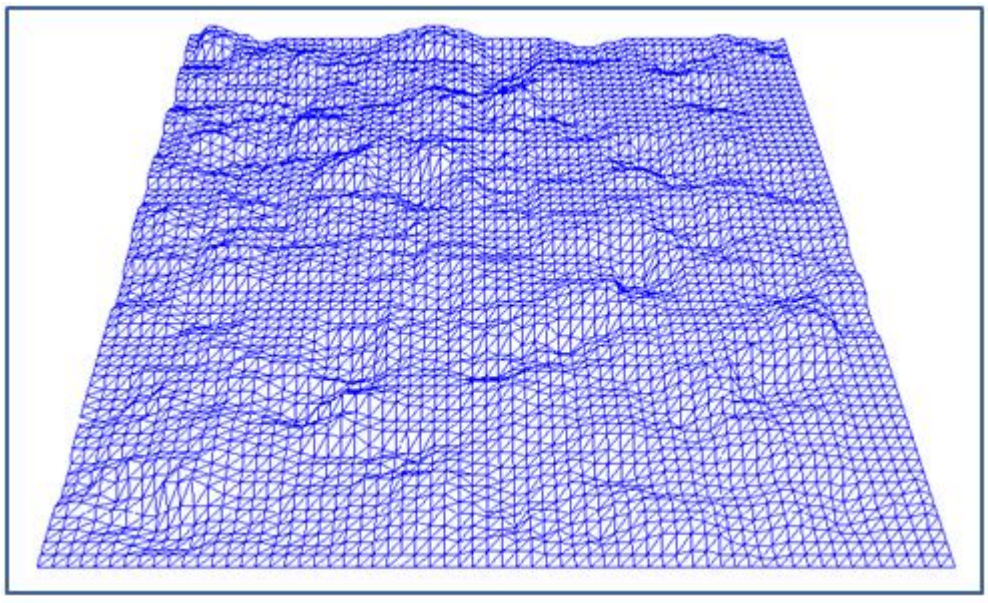
3.4. DÖRTGENSEL ALGORİTMA

Dörtgensel algoritması yukarıda anlatılan algoritmalarla benzer özellik taşımaktadır. Bu algoritmanın amacı diğer algoritmaların avantajlı olduğu noktaların birleştirilmesi ve ortaya daha kontrollü ve daha gerçekçi haritaların üretilmesini sağlamaktır. Algoritmanın temeli, arazi üzerine Şekil 3.5'de görüldüğü gibi dörtgensel prizmaların eklenmesiyle arazi modelini oluşturmaktır. Şekil 3.6 ve Şekil 3.7 farklı sayıda iterasyon sonucu oluşmuş arazi modelleri görülmektedir

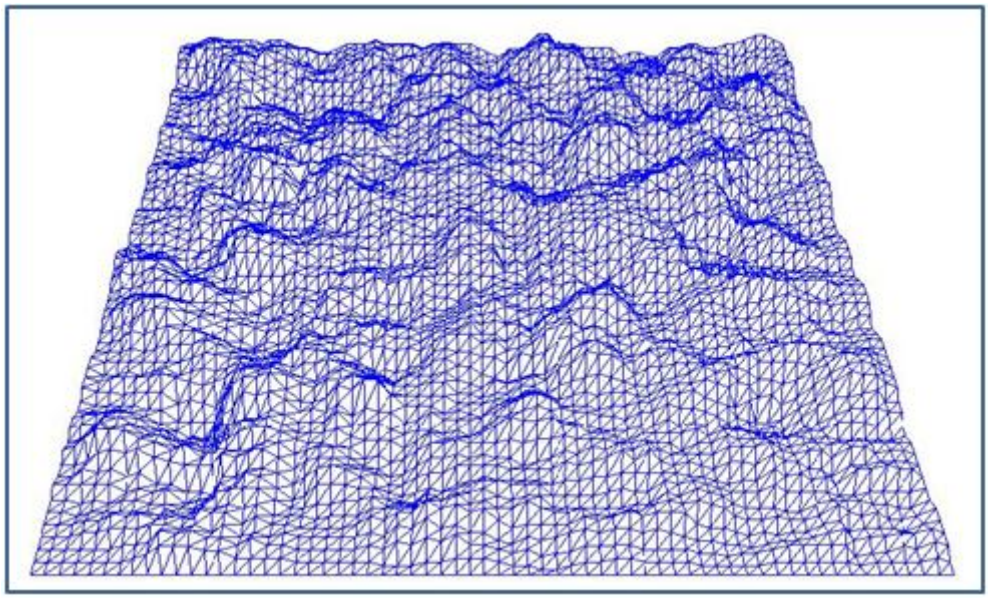
modelleri görülmektedir.



Şekil 3.5. Bir iterasyon sonucu oluşan arazi modeli.



Şekil 3.6. Üç yüz iterasyon sonucu oluşan arazi modeli.



Şekil 3.7. Bin iterasyon sonucu oluşan arazi modeli.

Çember algoritmasında iterasyon sayısı artıkça arazi üzerinde dalgalanmalar oluşmaktadır. Portakal kabuğunu andıran bu yapının oluşmaması için arazi üzerinde dörtgensel yükseltiler oluşturulmaktadır. Çember algoritmasındaki yarıçap avantajı bu algoritmada karşımıza çıkmaktadır. dörtgenin genişlik ve yükseklik değerleri kullanıcıdan alınarak kullanıcı etkileşimi sağlanabilmektedir.

Particle algoritmasındaki gibi keskin kenarlar elde etmek için oluşan dörtgenin orta noktası diğer noktaların yükseklik değerlerinden daha fazla artırılır. Böylece keskin kenarlar elde edilebilmektedir.

3.5. FFT ALGORİTMASI

FFT algoritmasının temeli 2D Fast Fourier Transform'a dayanmaktadır. Algoritma dört temel adımdan oluşmaktadır. İlk adımda *nxn* boyutlarındaki yükseklik matrisi belirlenir ve bu matrisin değerleri rastgele doldurulur. Daha sonra Bu matrisin 2D FFT dönüşümü sağlanır.
sağlanır.

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M} \sum_{y=0}^{N} f(x,y) e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \quad (3.6)$$

2D FFT formülü Eşitlik 3.6'da verildiği gibidir. Bu eşitlikte $f(x, y)$ x ve y noktasının yükseklik değerini, $F(u, v)$ ise FFT'si alınmış olan u,v noktalarının yeni değerini tutmaktadır. 2D FFT dönüşümü bize rastgele değerlerin frekansını verir. Yüksek frekanslar keskin geçişleri gösterdiğinden bu değerler yumuşatılır ve rastgele olarak belirlenmiş yükseklik değerler arasındaki keskin geçişler kaldırılmış olur. . Dördüncü ve son adımda ise 2D FFT'si alınmış matrisin ters FFT'si yani IFFT alınır. Bu işlemler sonucu ortaya daha yumuşak geçişli arazi yüzeyleri oluşur [24]. Ek Açıklamalar A, FFT algoritması ile üretilmiş arazi modellerinden örnekler sunmaktadır.

BÖLÜM 4

PARALEL PROGRAMLAMA

Eski yöntemlerde işlemcinin performansını artırmak için üzerinde bulunan transistor sayısı artırılarak çözüm üretilmeye çalışılmaktaydı. Yonga üzerinde bulunan transistor sayısının artması ile başka sorunları ortaya çıkarmaktadır. Isının yükselmesi, bu ısının dağılımı ve fazladan harcanan güç yaşanan problemler arasındadır. Bu problemleri çözüme kavuşturmak adına işlemci üreticileri yeni fikirler geliştirdiler. Geliştirilen yeni bir metot ile birlikte daha fazla performans sağlanmış oldu [25]. Bu metot ile birlikte çok işlemcili bilgisayarlar dönemine geçilmiş olundu.

Çok işlemcili bilgisayarlara geçilmesi bazı problemleri de beraberinde getirmiştir. Tek işlemcili bilgisayarlar için tasarlanmış programlar çok işlemcili bilgisayarlar üzerinde yeterli performansı sağlayamamaktadır. Bu sorunu çözmek adına çok işlemcili bilgisayarlara özel yeni bir teknik geliştirildi. Paralel hesaplama adı verilen bu teknik problemlerin ufak görev parçalarına bölünmesi ve bunların eş zamanlı olarak koordine edilmesine dayanır. Paralel hesaplama ile performans artmakta ve büyük sorunlar daha az sürede çözülebilmektedir.

4.1. GPGPU PROGRAMLAMA

Paralel programlama yöntemlerinin gelişmesine bağlı olarak üreticiler yeni fikir arayışları içine girmişlerdir. Bilgisayar işlemcisine ek olarak yeni kaynak arayışına girilmesi ile birlikte grafik işlemci birimleri (Graphics Processing Unit, GPU) kullanılmaya başlandı. Modern GPU'lar

- Üzerlerinde çok güçlü grafik motoru bulundurmaktadır,
- Aritmetik işlem yapabilme hızı bilgisayar işlemcisine oranla yüksektir,

- Hafıza band genişliği hızı bilgisayar işlemcisine oranla yüksektir,
- Üst seviyede paralel programlanabilir işlemciler barındırmaktadırlar [26].

3B uygulamalar için güçlendirilen GPU lar kendi aralarında da paralel programlanabilmektedir. Böylece tek bir bilgisayar üzerine birden fazla GPU eklenebilmektedir. SLI adı verilen bu teknik sayesinde özdeş ekran kartları paralel olarak bağlanarak işlem gücü arttırılır. Yüksek çözünürlükte 3B bilgisayar oyunları, mühendislik çizim uygulamaları, çeşitli eğitim simülatörleri gibi uygulamalarda ekran kartının performansı oldukça önemlidir. GPU'ların öneminin artması ile birlikte yeni bir teknik arayışında olan üreticiler GPGPU(General-Purpose computation on Graphics Processing Units)kavramını ortaya çıkarmışlardır [25]. Bu yöntemin amacı GPU'nun çalışmadığı durumlarda CPU gibi davranarak fazla iş yükünü almak ve performans kazancı sağlamaktır. GPGPU sayesinde çok yüksek kapasiteli bilgisayarlar oluşturmak mümkündür. Top500 listesindeki 3 süper bilgisayarın GPGPU tabanlı olması bunun ispatıdır [27].

4.1.1. NVIDIA

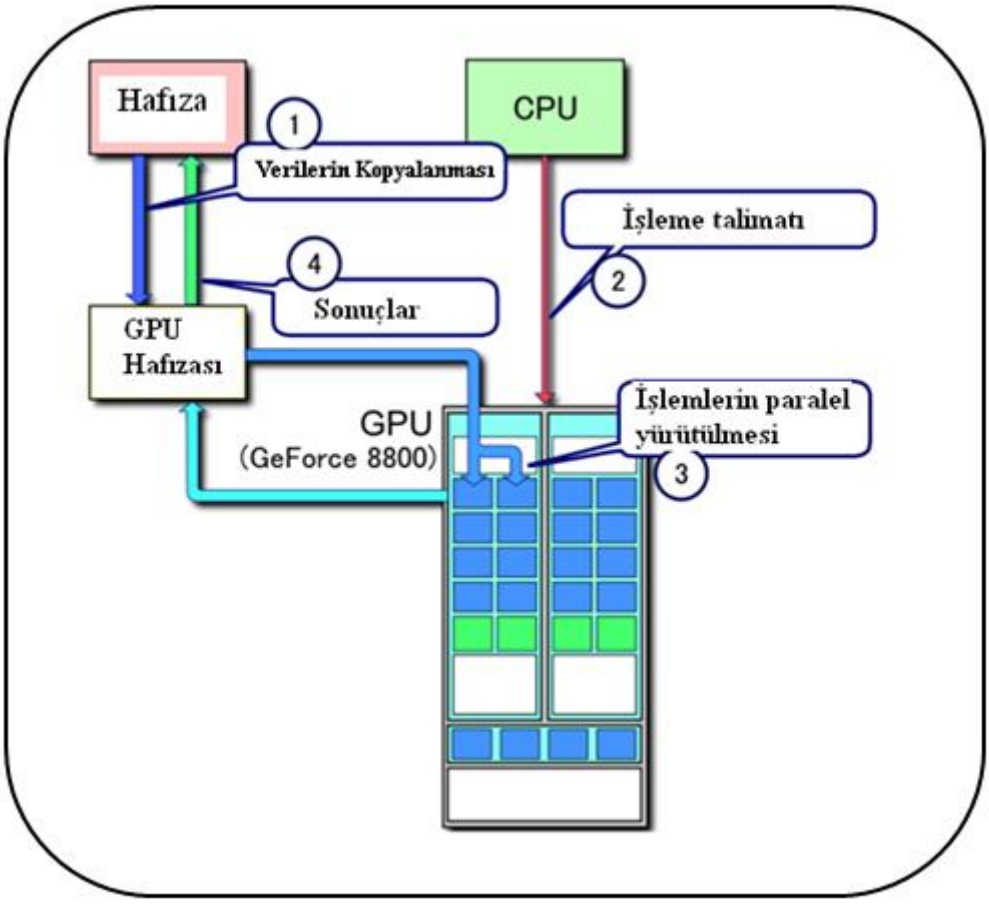
NVIDIA tarafından geliştirilen paralel işlem mimarisi CUDA' dır. CUDA yazılım geliştiriciler tarafından programlama dilleri ile birlikte kullanılan, NVIDIA için tasarlanmış bir işlem motorudur ve C derleyicisi ile derlenir. Python, Perl, Fortran, Java ve MATLAB ile birlikte kullanılabilmektedir.

CUDA yazılım geliştiricilere CUDA GPU içinde sanal komut setine ve paralel hesaplama birimlerinin hafızasına erişebilmeyi sağlar. CUDA ile NVIDIA GPU' lar işlemci gibi kullanılabilmektedir.

Bilgisayar oyunlarında sanal gerçeklik ortamı yaratmak için çeşitli fiziksel özellikler eklenir. Örneğin bir binanın patlaması, yangın çıkması yangın sırasında dumanın yükselmesi gibi özellikler sayesinde oyunlar ve simülatörler daha gerçekçi bir görünüme kavuşmuş olur. Bu fiziksel olayların hesaplanması beraberinde çok sayıda matematiksel işlemi gerektirmektedir. Hesaplamaların hızlı bir biçimde yapılıp efektlerin doğru bir şekilde kullanıcıya sunulması gerekmektedir. Bu amaçla CUDA

çeşitli matematiksel işlemleri paralel programlama mimarisi ile çok hızlı bir biçimde gerçekleştirmektedir. CUDA' nın sağladığı hızlı işlem yeteneği başka alanlarda da kullanılmasını sağlamıştır. Şifreleme, biyoloji, fizik hesaplamaları gibi bilimsel alanlarda da kullanılmaktadır [28–32].

CUDA hem yüksek seviyeli programlama hem de düşük seviyeli programlama yapmayı sağlar. İlk CUDA versiyonu 15 Şubat 2007 yılında piyasaya sürüldü. Bu versiyon Microsoft Windows ve Linux işletim sistemlerini destekliyordu. Daha sonraki sürümleri ile birlikte MAC OS işletim sistemi desteği sağlandı.



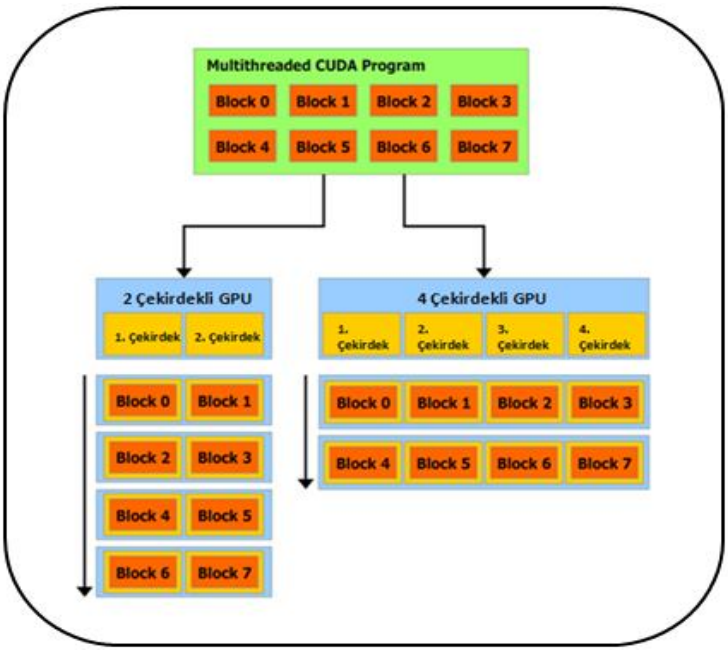
Şekil 4.1 CUDA işlem diyagramı [33].

Şekil 4.1 CUDA mimarisini göstermektedir. (1) durumda veriler bilgisayar belleğinden alınıp ekran kartı belleğine aktarılır. (2) durumda CPU tarafından görevler GPU ya devredilmiş olur. (3) durumda işlemlerin paralel bir şekilde yürütüldüğünü göstermektedir. Elde edilen sonuçlar tekrardan GPU hafızasından

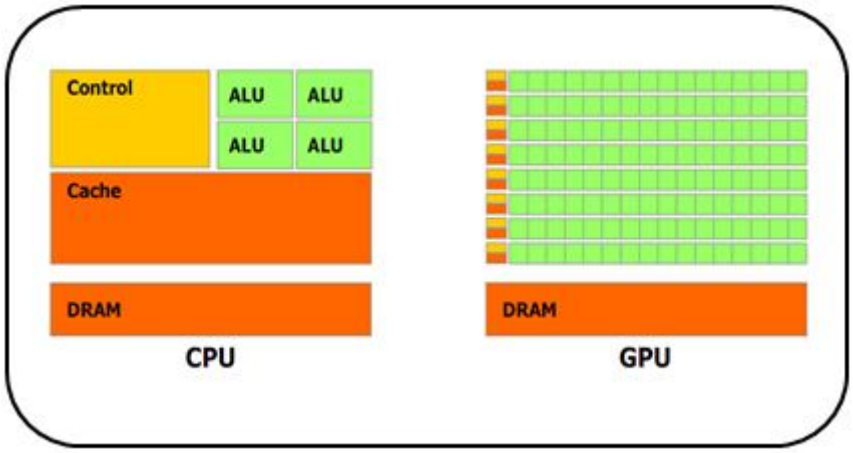
bilgisayar hafizasına iletir. Bu gösterim (4) durumda ok ile sembolize edilmiştir.

CUDA kullanımında verilerin hafızan alınıp GPU hafızasına alınması iş yükü olarak görülebilir. Bu taşıma esnasında zaman kayıpları yaşanmaktadır. Fakat çok büyük boyutlu matrislerde yapılan işlem hızı veri taşıma sırasında yaşanan zaman açığını kapatmakla kalmayıp CPU'ya oranla daha fazla performans artışı sağlamaktadır. CUDA'nın geliştirilme amacı çok büyük verilerde yaşanan zaman kayıplarını paralel programlayarak en aza indirmek olduğundan küçük boyutlu matrislerde CUDA kullanımı dezavantaj olmaktadır. Çünkü verilerin işlem hızı verilerin hafızadan taşınması sırasında harcanan zaman açığını kapatamayacaktır.

CUDA işlemleri birbirinden bağımsız işlem parçacıklarına böler. Buna aynı zamanda multithread programlama da denilmektedir. Multithread programlama büyük bir işlemin daha küçük işlem parçacıklarına bölünebilmesidir. CUDA sayesinde bu işlemler GPU üzerinde bulunan işlemciler tarafından gerçekleştirilmektedir. Şekil 4.2'de gösterildiği gibi GPU üzerinde bulunan işlemci sayısının artmasına bağlı olarak bölüştürülen iş parçacıkları bu işlemcilere dağıtılır. İşlemci sayısının artması az zamanda daha çok iş yapılabilmesini sağlar.

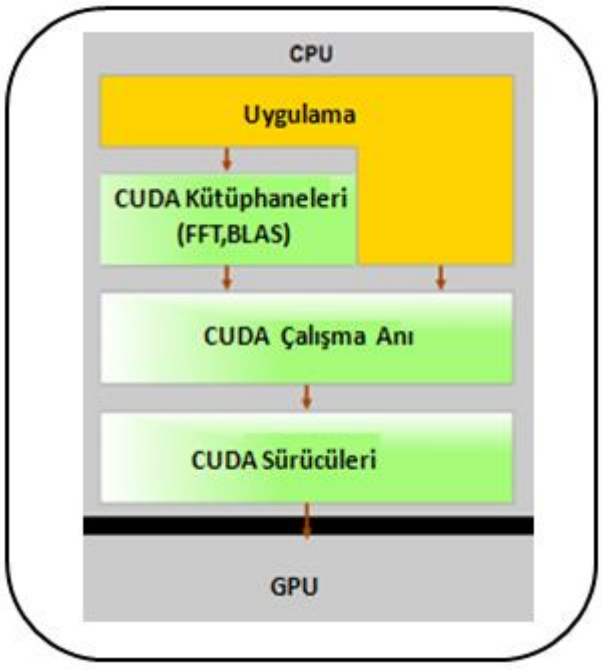


Şekil 4.2. CUDA Paralel programlama diyagramı [33].



Şekil 4.3 CPU ve GPU Yapısı [33].

GPU' nın normal bir CPU dan daha hızlı işlem yapabilme yeteneğinin nedenlerinden biri de Şekil 4.3'de gösterildiği gibidir. GPU grafik veri işlemleri için bünyesinde çok miktarda transistör ve ALU bulundurmaktadır. ALU aritmetik ve mantık işlemlerini gerçekleştiren sayısal bir devredir. ALU sayısının artması ile işlem gücü artmaktadır. CPU üzerinde ise kontrol birimleri ve ön bellek daha fazla bulunmaktadır.



Şekil 4.4 CUDA uygulama mimarisi [33].

Bir CUDA uygulaması incelendiğinde Şekil 4.4'de gösterildiği gibi işlemler CPU üzerinden başlar. CPU ya gönderilen komutlar sayesinde uygulama başlatırlır. Uygulama içerisinde CUDA'nın çeşitli kütüphaneleri kullanılabilmektedir. CUDA komutları derlenir ve gerekli matematiksel hesaplamalar CUDA sürücüleri aracılığıyla GPU üzerinden gerçekleştirilir.

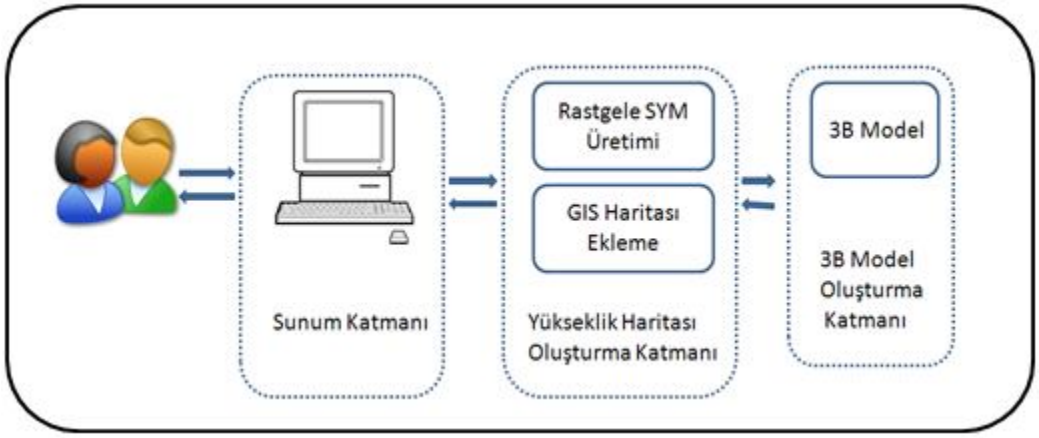
4.1.2. ATI

ATI işlemcileri CUDA programlamayı desteklemezler. ATI ekran kartları GPGPU sağlamak için kendi standardını geliştirmiştir. Bu amaçla AMD APP adını verdikleri yeni teknolojilerini duyurdular. AMD APP OPENCL desteği sağlamaktadır. Tıpkı CUDA gibi ATI GPU' ları da paralel olarak programlanabilmektedir. OPENCL hem NVDIA hem de ATI GPU' ları ile birlikte kullanılan bir GPGPU tekniğidir. Bu sayede heterojen sistemler paralel olarak programlanabilmektedir [34].

BÖLÜM 5

UYGULAMANIN GELİŞTİRİLMESİ

Geliştirilen uygulama sayesinde yükseklik haritaları gerçek zamanlı olarak oluşturulabilmekte ve düzenlenebilmektedir. Şekil 5.1'de gösterildiği gibi uygulama 3 katmandan oluşmaktadır. İlk katman kullanıcıların uygulama ile etkileşiminin sağlandığı sunum katmanıdır. Bu katman bilgisayar yardımı ile sağlanmaktadır. Sunum katmanında kullanıcı 3B modeller oluşturabilir ve model üzerinde değişiklikler yapabilir. İkinci katman olan yükseklik haritası oluşturma katmanı kullanıcıdan gizlenmiş bir vaziyettedir. Bu katmana ancak sunum katmanı aracılığıyla erişilebilir. Bu katmanda çeşitli rastgele yükseklik haritası oluşturma algoritmaları kullanıldığı gibi çeşitli GIS harita formatları da işlenerek nxn boyutlarında bir yükseklik matrisi elde edilir. Elde edilen bu yükseklik matrisi model oluşturma katmanına gönderilir. Bu katman almış olduğu matrisi OpenGL tabanlı bir grafik motoru aracılığıyla 3B olarak modeller. Hazırlamış olduğu modeli sunum katmanına gönderir.



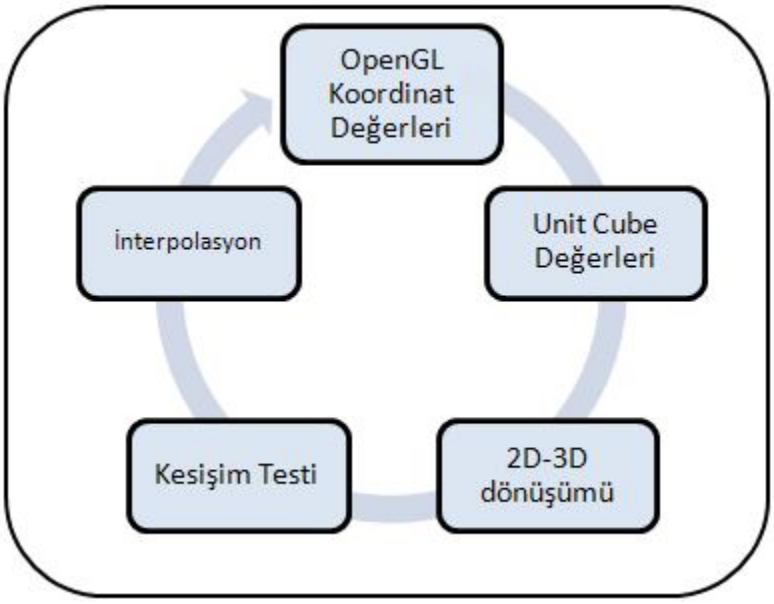
Şekil 5.1. Sistem mimarisi.

Geliştirilen yazılımda programlama dili olarak Java kullanılmıştır. Javanın nesne tabanlı bir dil olması sistemin katmanlı bir biçimde modellenmesini sağlamıştır.

Böylece her bir katmanda birbirinden bağımsız değişiklikler sorunsuz bir biçimde gerçekleştirilebilmektedir. Sistemin geliştirilmesi esnekleştirilmiştir. Java'nın platform bağımsız olması uygulamanın her ortamda kullanılmasını sağlayacaktır.

5.1. KULLANILAN METOTLAR

Yükseklik haritalarının modellenmesi ve bu model üzerinde çeşitli değişikliklerin yapılabilmesi için gerekli olan işlemler Şekil 5.1'de gösterildiği gibi bir döngü şeklinde sürekli tekrar edilmektedir.

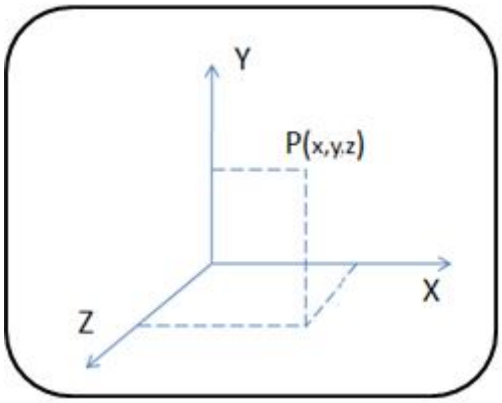


Şekil 5.2 Uygulama döngüsü.

Bu döngünün her adımda birbirini izlemesi gerekmektedir. Model üzerinde herhangi bir modifikasyon işlemleri yapılmadığı müddetçe 2D-3D dönüşümü, kesişim testi ve interpolasyon adımlarının gerçekleştirilmesine ihtiyaç duyulmamaktadır

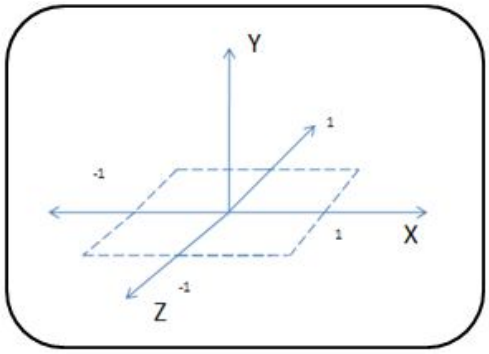
5.1.1. OpenGL Koordinat Değerlerinin Hesaplanması

Uygulamanın ilk adımı yükseklik haritasının 3 boyutlu bir modelinin oluşturulması aşamasıdır. 3 boyutlu düzlemde x, y ve z olmak üzere üç çeşit eksen bulunmaktadır. Bu eksenlerin gösterimi Şekil 5.3'de gösterilmektedir.



Şekil 5.3 3B koordinat düzlemi.

Yükseklik modeli oluşturulurken arazi üzerindeki her bir noktanın 3 boyutlu koordinat düzleminde hangi noktalara denk geleceği hesaplanmalıdır. Arazi üzerindeki herhangi bir noktanın y eksenine karşılık gelen değeri o noktanın yükseklik değerini vermektedir. 3 boyutlu koordinat düzlemi sonsuza doğru gittiğinden arazi modelinin yerleştirileceği alan belirlenir. Şekil 5.4'de gösterildiği gibi oluşturulan arazi modeli X ekseninde [-1,1] ve Z ekseninde [-1,1] de sınırlandırılmıştır. Bu çalışmada arazi modeli üzerindeki her bir noktanın x ve z koordinat değerleri Eşitlik 5.1 ve Eşitlik 5.2 aracılığıyla hesaplanmıştır.



Şekil 5.4 Arazi modelinin 3B yeri.

$P(x_i,y_i,z_i)$ Arazi üzerindeki herhangi bir noktadır.

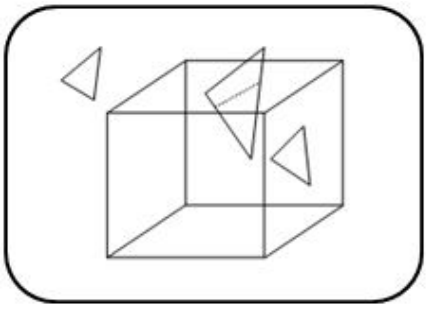
$$x_i = x_0 + d_x * u \tag{5.1}$$

$$z_i = z_0 + d_z * v \tag{5.2}$$

Arazinin yerleştirileceği X eksenin başlangıç değeri x_0 ve x_0+d_x ise bitiş değeridir. Bu formül aracılığıyla arazi üzerindeki her bir noktanın X ekseni üzerinde bulunan değerleri [x_0 , x_0+d_x] arasında olacaktır. Diğer bir değer olan u değeri aracılığıyla noktalar X ekseni üzerinde belirli aralıklarla yerleştirilmektedir. Aynı şekilde z_0 değeri arazinin yerleştirileceği Z eksenin başlangıç değeri z_0 ve z_0+d_z bitiş değeridir. Bu formül aracılığıyla arazi üzerindeki her bir noktanın Z ekseni üzerinde bulunan değerleri [z_0 , z_0+d_z]arasında olacaktır. Diğer bir değer olan v değeri aracılığıyla noktalar Z ekseni üzerinde belirli aralıklarla yerleştirilmektedir. P noktasının diğer bir eksen değeri olan y_i o noktanın yükseklik değeridir. Her bir nokta için bu hesaplamalar sonucu Arazi modelinin 3 boyutlu koordinat değerleri tespit edilmiş olur. Bu şekilde noktaların alan üzerine yerleştirilmesine mozaikleştirme (tessellation) denir.

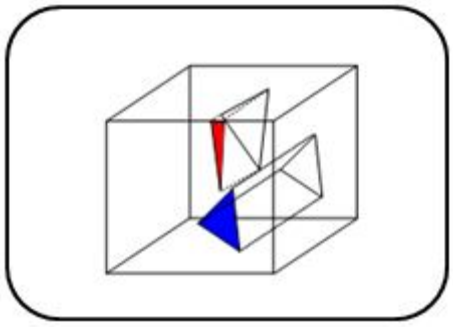
5.1.2. Birim Küp (Unit Cube) Değerlerinin Hesaplanması

Birim Küp (Unit Cube) OpenGL kütüphanesinin kullandığı sanal bir yapıdır. Koordinat değerlerini verdiğimiz nesnelerin birim küp değerleri OpenGL tarafından tespit edilir ve bu küp içerisine yerleştirilir. Şekil 5.5'de gösterildiği gibi nesnelerin koordinat değerleri verilmiştir. Bu nesnelerin birim küp değerleri OpenGL tarafından otomatik olarak tespit edilmiş ve yerleştirilmiştir.



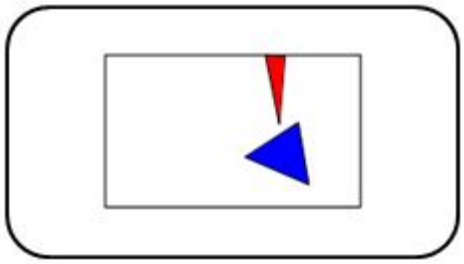
Şekil 5.5. Birim küp (unit cube) gösterimi.

Koordinat değerleri verilen nesneleri görebilmemiz için OpenGL kütüphanesine, bu birim küpe nereden baktığımızın bilgisi verilmektedir. Şekil 5.6 birim küpün hangi tarafından bakıldığını göstermektedir.



Şekil 5.6. Birim küpe bakış.

Bu işleme projeksiyonun ayarlanması da denilebilir. Bu işlem ile birim küp dışında kalan bütün nesneler kırpılmış olur. Kırpma işleminin ardından nesneler ekranda Şekil 5.7'deki gibi gösterilir. Birim küp dışında kalan hiçbir nesne ekranda gösterilmemiş olur.



Şekil 5.7. Ekran çıktısı.

Birim küp değerlerinin tespit edilebilmesi için OpenGL kullanmış olduğu Projeksiyon matrisi ve ModelView Matrisinin bilinmesi gerekmektedir. OpenGL de bu matris değerlerini almak için aşağıdaki kodlardan faydalanılır.

- glGetFloatv(GL_MODELVIEW_MATRIX,…)
- glGetFloatv(GL_PROJECTION_MATRIX,…)

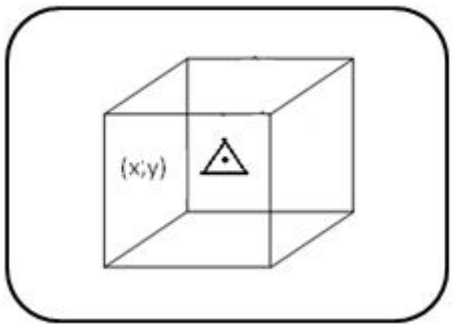
$$V_{unit}\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = P\begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} * M\begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{bmatrix} * V\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} \quad (5.3)$$

Eşitlik 5.3'de gösterildiği gibi V arazi üzerinde herhangi bir noktayı temsil etmektedir. Bu noktanın koordinat değerleri öncelikle M olan Modelview matrisi ile çarpılmakta elde edilen değerler P olan projeksiyon matrisi ile çarpılmaktadır. Çarpımlar sonucundan V_{unit} Birim küp matrisi elde edilmiş olur.

5.1.3. 2D-3D Dönüşümü

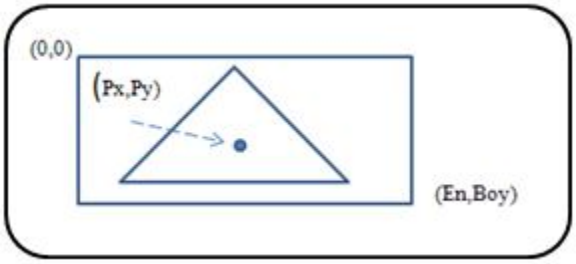
3 boyutlu modeller bilgisayar ekranlarında 2 boyutlu olarak sunulmaktadır. Bilgisayar ekranı üzerinde seçilen herhangi bir noktanın koordinat değerleri 2 boyutlu olduğundan bu noktanın 3 boyutlu model üzerinde hangi noktaya denk geldiğinin tespit edilmesi gerekmektedir. Birim küp içerindeki bir noktanın gösterimi Şekil 5.8'de gösterilmiştir. Bu noktanın x ve y koordinat değerleri (x,y) şeklinde gösterilmektedir.

.



Şekil 5.8. Birim Küp içerisinde nesne gösterimi.

Bu noktanın 2 boyutlu ekran üzerindeki görüntüsü Şekil 5.9'da gösterilmiştir. Bilgisayar ekranlarının sol üst köşesi (0,0) koordinat değerlerine sahiptir. Ekranın genişlik ve yüksekliğine bağlı olarak sağ alt köşesinin koordinat değerleri değişmektedir. Ekranın koordinat değerlerindeki bu değişiklik ekranda gösterilen nesnelerinde değerlerini etkilemektedir. 3 Boyutlu noktanın ekran üzerindeki yeni koordinat değerleri Eşitlik 5.4 ve Eşitlik 5.5'de verilmiştir.



Şekil 5.9. 3B düzlemin 2B düzlemde görüntüsü.

$$P_x = \left(\frac{x+1}{2}\right) * En \qquad P_x \epsilon [0, En] \tag{5.4}$$

$$P_y = \left(\frac{1-y}{2}\right) * Boy \qquad P_y \epsilon [0, Boy] \tag{5.5}$$

Eşitlik 5.4 ve Eşitlik 5.5'da verilen dönüşümlerin tam tersi bize 2 boyutlu ekran üzerinde seçilen noktanın 3 boyutlu model üzerinde hangi noktaya geldiğini bulmamızı sağlar. Eşitlik 5.6 ve Eşitlik 5.7 ekran üzerinde seçilen bir noktanın 3 boyutlu koordinat değerlerinin hesaplanmasını göstermektedir.

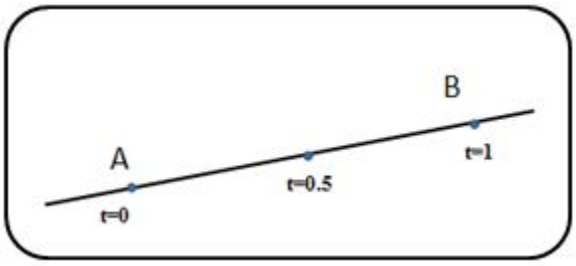
$$x = \left(-1 + \frac{2P_x}{En}\right) \quad x \in [-1,1] \tag{5.6}$$

$$y = \left(1 - \frac{2P_y}{Boy}\right) \quad y \in [-1,1] \tag{5.7}$$

Fakat bu yeterli bir dönüşüm sağlayamaz. Çünkü bu dönüşümlerde derinlik değeri olan Z koordinat ekseni kullanılmadı. Bunun için 2 adet nokta belirlenir ve bu iki noktanın Z değerlerine 1 ve -1 değeri verilir. Eşitlik 5.8 Z koordinat değerine -1 verilmesi ile oluşan A noktasını gösterirken Eşitlik 5.9 Z koordinat değerine 1 verilen B noktasını göstermektedir. Elde edilen yeni noktalardan Şekil 5.10'da gösterildiği gibi bir doğru parçası elde edilir.

$$A = (x, y, -1) \tag{5.8}$$

$$B = (x, y, 1) \tag{5.9}$$



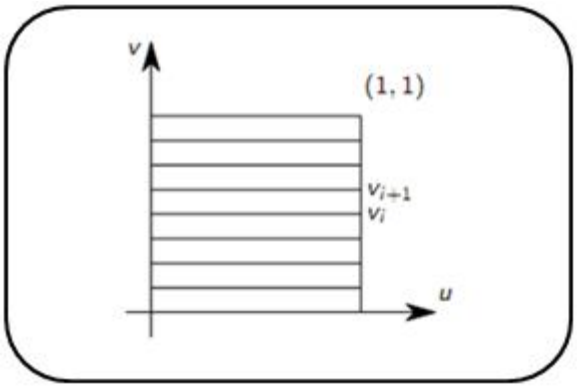
Şekil 5.10. [AB] Doğru parçası.

$$P(t) = A + t(B - A) \;\; t \in R \; ve \; P, A, B \in R^3 \tag{5.10}$$

Eşitlik 5.10, A ve B noktaları arasında kalan doğru parçası üzerindeki her bir noktanın t değişkenine bağlı formülünü göstermektedir. Formüldeki t değerinin 0,5 olması sonucu nokta doğru parçasının tam ortasındaki noktayı, 0 olması A noktasını ve 1 olması B noktasını vermektedir. 3 boyutlu model üzerinde kalan bütün noktalar [AB] ile kesişim testinden geçirilir ve bu doğru parçası ile kesişen nokta bize 2 boyutlu ekran üzerinden hangi noktayı seçtiğimizi verir. Kesişim testi için Barycentric Koordinat Düzleminden faydalanılmaktadır.

5.2. MOZAİKLEŞTİRME (TESSELLATION)

Tessellation Türkçedeki sözlük anlamı mozaikleştirme olarak geçmektedir Mozaikleştirme bir yüzeyin boşluk bırakmayacak şekilde poligonlara bölünmesidir. Bilgisayar grafiklerinde en çok üçgen kullanılmaktadır. Poligon sayısının artmasına bağlı olarak temsil edilen yüzey üzerindeki detaylar artmaktadır. Tessellation yardımı ile arazi modeli 3B olarak modellenirken elde edilen yükseklik noktalarının arazi üzerine yerleştirilmesinde kullanıldı. Bu amaçla [0,1]x[0,1] boyutlarında temel bir alan oluşturuldu. Arazi önce u ve v koordinatlarına yerleştirilerek modellenmesi yapıldı. Daha sonra yükseklik noktalarının arazi üzerine yerleştirilmesi için bu dörtgen üzerine iki boyutlu Tessellation uygulandı. Şekil 5.11 yüzeyin yatay olarak parçalandığını göstermektedir.

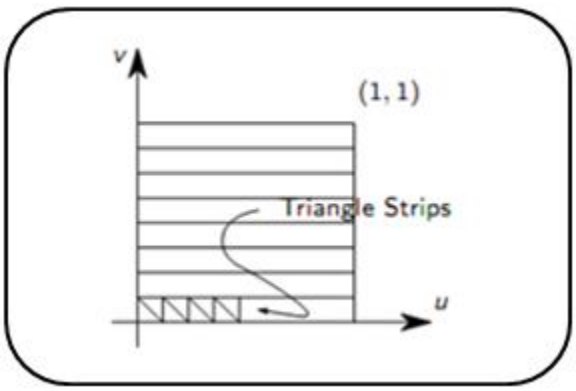


Şekil 5.11. u ve koordinatları sahip alanın bölünmesi [35].

Şekil 5.11'de verilen mozaikleştirme işleminin sembolik kodu aşağıda verilmiştir.

```
N:Toplam Nokta Sayısı
float step=1/N;
for( int i=0;i<N;i++){
    vi=i*step;
    vi+1=vi+step;
//Yapılan işlemler
}
```

3D modellemede bu işlem yeterli olmayacağından şeritler halinde bölünmüş parçaları üçgenler şeklinde bölmek gerekmektedir. OpenGL de bunun için GL_TRIANGLE_STRIP kullanılmaktadır. Şekil 5.12'deki gibi tek bir şeridin üçgenlerle doldurulmasını sembolize eder.



Şekil 5.12. Şeritlere bölünmüş alanın üçgenlerle doldurulması [35].

Yapılan işlemin sembolik kodu aşağıda verilmiştir.

```
glBegin(GL_TRIANGLE_STRIP)
for(int j=0;j>N;j++){
    u_j = jxstep;
    glVertex3f(getX(u_j, v_j), getY(u_j, v_j), getZ(u_j, v_j));
    glVertex3f(getX(u_j, v_j+i),getY(u_j, v_j+i), getZ(u_j, v_j+i));
}
```

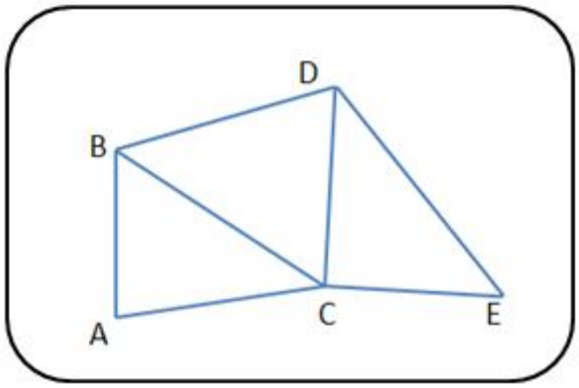
Şekil 5.4 sadece bir şeridi üçgenlerle doldurduğundan yeterli olmamaktadır. Tüm alanı üçgenlerle doldurmak için aşağıda sembolik kodu verilmiş yapı kullanılır.

```
    N: Toplam nokta sayısı
    float step=1/N;
    for(int i=0;i<N;i++){
        vi=i*step;
        vi+1=vi+step;
        glBegin(GL_TRIANGLE_STRIP)
        for(int j=0;j<N;j++){
            uj=j*step;
                glVertex3f(getX(u_j, v_j), getY(u_j, v_j), getZ(u_j, v_j));
                glVertex3f(getX(u_j, v_j+i),getY(u_j, v_j+i), getZ(u_j, v_j+i));
            }
        glEnd();
    }
```

Burada her bir şeride bulunan toplam üçgen sayısı $2xN$, yüzey üzerinde bulunan toplam şerit sayısı N ve üçgen sayısı $2xN^2$ olacaktır.

5.3. BARYCENTRIC KOORDİNAT DÜZLEMİ

Bilgisayar grafiklerinde nesneler genelde üçgenler yardımıyla modellenmektedir. 3 boyutlu modellenmek istenen nesne üçgenlerin uç uca birleşmesinden oluşur. Bu yapıyı OpenGL de Triangle Strip adı verilen yapıyla bunu sağlamak mümkündür. Triangle Strip ile aynı noktayı paylaşan üçgenler oluşturulmuş olur. Şekil 5.13'de gösterildiği gibi üçgenler birleştirilerek her hangi bir nesne modeli gerçekleştirilir.



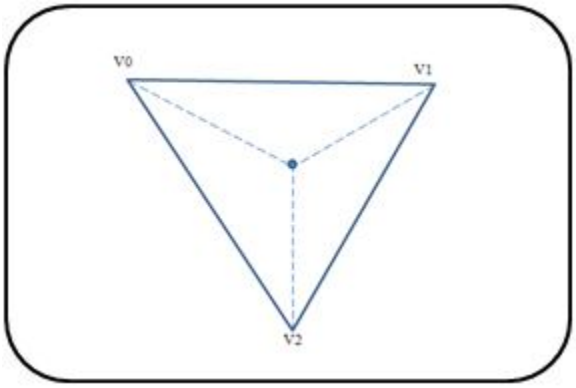
Şekil 5.13. TriangleStrip ile oluşmuş üçgenler.

Triangle Strip de ilk üçgen için noktalar verilir. Daha sonra her bir nokta daha önceki noktalarla birleştirilerek üçgen oluşması sağlanır. Şekil 5.13'de verilen A, B ve C noktalarından sonra gelen D noktası kendinden önceki iki noktayla yeni bir üçgen oluşturur. Daha sonra gelen E noktası kendinden önceki iki nokta olan C ve D noktaları ile bir üçgen oluşturmuş olur. Bu sayede kod verimliliği ve performans artar. Örneğin N tane üçgen tanımlamak için $N * 3$ adet nokta vermek gerekirken Triangle strip ile $N + 2$ adet nokta verilmesi yeterli olur. Bu sayede daha az hafıza harcanmış olur.

```
glBegin(GL_TRIANGLE_STRIP);
    glVertex3f( 0.0f, 0.0f, 0.0f ); //1. Nokta
    glVertex3f( 0.0f, 1.0f, 0.0f ); //2. Nokta
    glVertex3f( 1.0f, 0.0f, 0.0f ); //3. Nokta
   glVertex3f( 1.5f, 1.0f, 0.0f ); //4. Nokta
   glVertex3f( 1.5f, 1.0f, 0.0f ); //5. Nokta
```

glEnd();

Yukarıda OpenGL kod parçacığı ile Şekil 5.13 çizdirilebilir. Üçgenlerin bilgisayar grafiklerinde yaygın olarak kullanılması sonucu Barycentric Koordinat düzlemi önem kazanmaktadır. Bu koordinat düzlemi ile bir üçgen içerisindeki herhangi bir noktanın değeri üçgeninin köşe noktalarının kullanılması ile bulunabilir. Üçgenin bir köşesinden başlayarak kenarları belli katsayılarla çarpılıp vektörel olarak bu köşeye eklendiğinde üçgen içinde istenilen noktaya gitmek mümkündür. Kenarları çarpmada kullanılan katsayılara barycentric koordinatlar denir.

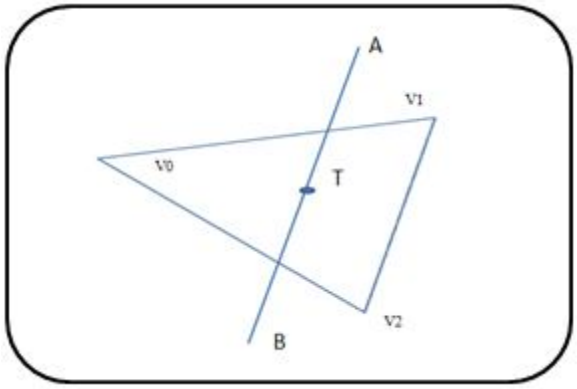


Şekil 5.14. Barycentric koordinat düzlemi.

$$V(u,v) = V_0 * (1-u-v) + V_1 * u + V_2 * v \quad (5.11)$$

$$u, v \geq 0,\ (1-u-v) \geq 0\ ve\ u+v \leq 1$$

Şekil 5.14'de verilen üçgenin köşe noktaları V_0, V_1, V_2 'dir. Üçgen içerisinde $V(u,v)$ noktasının yeri barycentric koordinatlar cinsinden nasıl hesaplandığına dair vektörel ifade Eşitlik 5.11'de verilmiştir [36]. Barycentric Koordinat düzleminden faydalanılarak kesişim testi yapılabilir.



Şekil 5.15. [AB] doğru parçasının üçgen ile kesişimi.

Şekil 5.15 [AB] doğru parçasının V_0, V_1, V_2 noktalarına sahip üçgen içerisinden geçtiğini göstermektedir. [AB] doğru parçasının üçgen içerisinden geçtiği T noktasının yeri bulunabilmektedir.

$$A + t(B - A) = V_0 * (1 - u - v) + V_1 * u + V_2 * v \tag{5.12}$$

$$t \in [0,1] \qquad u, v \geq 0 \qquad u + v \leq 1$$

Eşitlik 5.10 ve Eşitlik 5.11 den faydalanılarak Eşitlik 5.12 yazılabilir. Eşitlik 18'in sağlanması sonucu bulunan t değeri [AB] doğru parçasının bu üçgen ile kesiştiğini verir. Ayrıca kesişme noktası olan T noktasının yerini de vermiş olur. Eşitlik 18'de verilen noktalar 3 boyutlu olduğundan bu denklemin çözülebilmesi için Gauss Eliminasyon yönteminden faydalanılır.

5.4. GAUSS ELİMİNASYON METODU

Gauss Eliminasyon yöntemi denklem sistemlerinin çözümünde kullanılan bir yöntemdir. İki aşamadan oluşmaktadır. Birinci aşamada üst üçgen matrisi oluşturulur. İkinci aşamada ise geriye doğru sarma yöntemi ile bilinmeyen değerler bulunur. Uygulamada Eşitlik 5.12'de verilen denklemin çözülmesi için gauss eliminasyon metodundan faydalanılmıştır. Eşitlik 5.12'de A ve B değerleri ile V_0,V_(1) ve V_2 değerleri bilinmektedir. Bu değerler 3 boyutlu olduğundan

Eşitlik 5.12' in düzenlenmesi sonucu Eşitlik 5.13.'de gösterilen matris haline getirilmiştir [37].

$$(1-u-v)*\begin{bmatrix}V_{0x}\\V_{0y}\\V_{0z}\end{bmatrix}+u*\begin{bmatrix}V_{1x}\\V_{1y}\\V_{1z}\end{bmatrix}+v*\begin{bmatrix}V_{2x}\\V_{2y}\\V_{2z}\end{bmatrix}=(1-t)*\begin{bmatrix}A_x\\A_y\\A_z\end{bmatrix}+t*\begin{bmatrix}B_x\\B_y\\B_z\end{bmatrix} \quad (5.13)$$

$$\begin{bmatrix}\begin{bmatrix}V_{1x}-V_{0x}\\V_{1y}-V_{0y}\\V_{1z}-V_{0z}\end{bmatrix}\begin{bmatrix}V_{2x}-V_{0x}\\V_{2y}-V_{0x}\\V_{2z}-V_{0x}\end{bmatrix}\begin{bmatrix}A_x-B_x\\A_y-B_y\\A_z-B_z\end{bmatrix}\begin{bmatrix}u\\v\\t\end{bmatrix}\end{bmatrix}=\begin{bmatrix}A_x-V_{0x}\\A_y-V_{0y}\\A_z-V_{0z}\end{bmatrix} \quad (5.14)$$

Eşitlik 5.13. düzenlenmesi sonucu Eşitlik 5.14'de gösterilen matris oluşturulur. Bu matrisin gauss eliminasyon yöntemi ile çözülebilmesi için katsayılar matrisine dönüştürülmesi gerekmektedir. Eşitlik 5.15 ile katsayılar matrisi oluşturulmuştur.

$$\begin{bmatrix}\begin{bmatrix}V_{1x}-V_{0x}\\V_{1y}-V_{0y}\\V_{1z}-V_{0z}\end{bmatrix}\begin{bmatrix}V_{2x}-V_{0x}\\V_{2y}-V_{0x}\\V_{2z}-V_{0x}\end{bmatrix}\begin{bmatrix}A_x-B_x\\A_y-B_y\\A_z-B_z\end{bmatrix}\begin{bmatrix}A_x-V_{0x}\\A_y-V_{0y}\\A_z-V_{0z}\end{bmatrix}\end{bmatrix} \quad (5.15)$$

Önce birinci sütundaki köşegen elemanını 1 yapacak şekilde tüm satır uygun katsayı ile çarpılır. Sonra ilk satır uygun değerler ile çarpılarak ikinci ve üçüncü satıra eklenir ve bu satırlardaki 1. Sütunun elemanları 0 olur. Eşitlik 5.16'da gösterilmiştir.

$$\begin{bmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\0 & a_{22} & a_{23} & a_{24}\\0 & a_{32} & a_{33} & a_{34}\end{bmatrix} \quad (5.16)$$

Aynı işlemler diğer sütunlar içinde gerçekleştirildikten sonra Eşitlik 5.17'deki gibi üst üçgen matrisi elde edilmiş olur.

$$\begin{bmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\0 & a_{22} & a_{23} & a_{24}\\0 & 0 & a_{33} & a_{34}\end{bmatrix} \quad (5.17)$$

Üst üçgen matrisi elde edildikten sonra geriye sarma ile bilinmeyen u, v ve t değerleri bulunur.

$$t = a_{34}/a_{33} \tag{5.18}$$

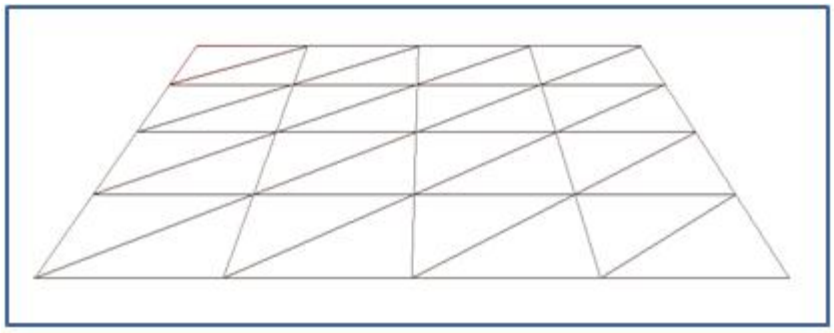
$$v = (a_{24} - a_{23} * t)/a_{22} \tag{5.19}$$

$$u = (a_{14} - (a_{13} * t + (a_{12} * v))/a_{11} \tag{5.20}$$

Eşitlik 5.18. bize t değerini verir. Bulunan t değeri bir üst satırda yerine konularak Eşitlik 5.19.'de v değeri bulunur son adım olarak Eşitlik 5.20'de gösterildiği gibi bulunan v ve t değerleri yerine konularak u değeri elde edilir. Gauss metodu sonucu bulunan t değeri bize 3B arazi modeli üzerindeki hangi noktanın seçildiği bilgisini vermektedir. Eşitlik 5.16'da yerine konulan t değeri ile bulunan noktanın yükseklik verisi gerçek zamanlı artırılabilmektedir.

5.5 ARAZİ MODELİNİN ÇÖZÜNÜRLÜĞÜNÜN ARTIRILMASI

Sayısal yükseklik haritası üzerinde yapılan düzenleme işlemleri sonucunda arazi boyutu artmaktadır. Şekil 5.16'da gösterildiği gibi 5x5 boyutundaki bir arazi modeli üzerinde düzenleme işlemi sonucunda elde edilen yeni arazi Şekil 5.17'de gösterilmektedir.



Şekil 5.16. *5x5* boyutundaki yükseklik haritası.

Arazi boyutundaki değişiklikler Eşitlik 5.21 yardımı ile hesaplanabilmektedir.

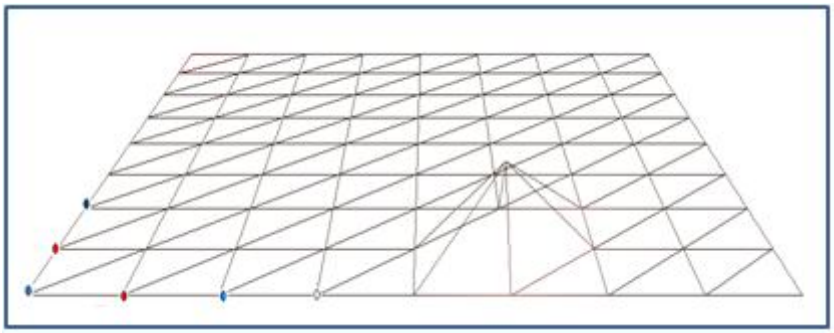
$$D_{new} = (R * (D_{old} - 1) + 1) \quad (5.21)$$

- D_{new} : Düzenleme sonucunda oluşan arazinin yeni boyutu.
- D_{old} : Arazinin ilk boyutu.
- R : Kullanıcı tanımlı çözünürlük değeri.

Şekil 5.16'da gösterilen *5x5* boyutundaki arazi için çözünürlük değeri 2 seçildiğinde oluşacak olan yeni arazinin boyutunun hesaplanması Eşitlik 5.22'de verilmiştir.

$$D_{new} = (2 * (5 - 1) + 1) \quad (5.22)$$

$$D_{new} = 9$$

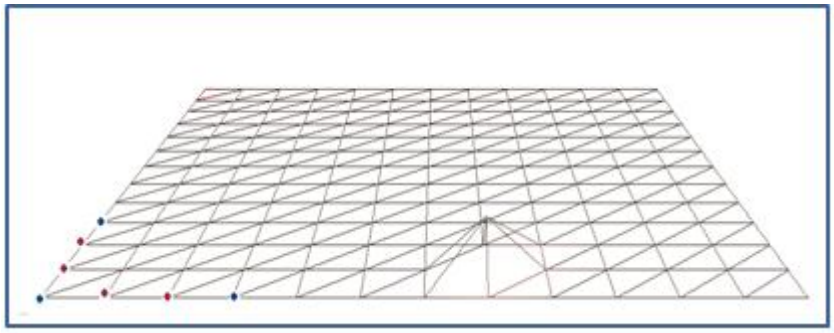


Şekil 5.17. Yükseklik haritasının R=2 ile düzenlenmesi.

Yapılan hesaplamalar sonucunda oluşan arazinin yeni görüntüsü Şekil 5.17'de gösterilmektedir. Yapılan düzenleme öncesinde tanımlı noktaların yükseklik değerleri aynı şekilde korunmakta ve yeni oluşan değerlerin yükseklik değerleri bilineer interpolasyon aracılığıyla hesaplanmaktadır. Çözünürlük değeri olan R değerinin 3 olması sonucunda oluşacak olan arazinin formülü Eşitlik 5.23 ile hesaplanabilmektedir.

$$D_{new} = (3 * (5 - 1) + 1) \quad (5.23)$$

$D_{new} = 13$

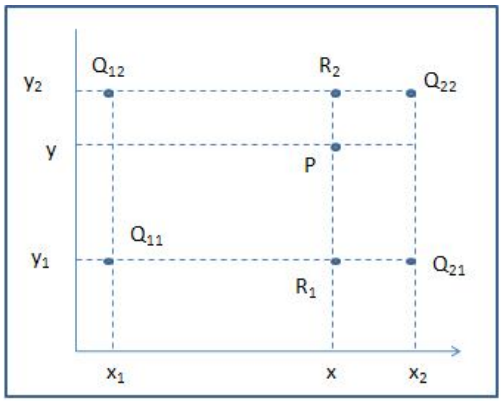


Şekil 5.18. Yükseklik haritasının R=3 ile düzenlenmesi.

Şekil 5.18'de görüldüğü gibi oluşan yeni arazi modeli $13x13$ boyutundadır. Çözünürlük değerinin artması sonucu arazi boyutu ve çözünürlüğü artış göstermektedir.

5.6. BİLİNEER İNTERPOLASYON

Bilinear interpolasyon sözlük anlamı çift doğrusal ara değerleme olarak geçmektedir. Ara değer üretmek için kullanılır. Ara değer üretmek için etrafında bulunan dört adet noktanın değerlerinden faydalanılır. Şekil 5.19'daki P istenilen nokta olsun. P'nin değerini hesaplamak için Q_{11}, Q_{12},Q_{21},Q_{22} nin değerlerinin bilinmesi gerekmektedir.



Şekil 5.19 Bilinear interpolasyon.

$$f(x,y) \approx \frac{f(Q11)}{(x2-x1)(y2-y1)}(x2-x)(y2-y) + \frac{f(Q21)}{(x2-x1)(y2-y1)}(x-x1)(y2-y) + \frac{f(Q12)}{(x2-x1)(y2-y1)}(x2-x)(y-y1) + \frac{f(Q22)}{(x2-x1)(y2-y1)}(x-x1)(y-y1) \quad (5.24)$$

Sayısal yükseklik haritası üzerinde yapılan düzenleme işlemleri sonucunda arazi boyutu artmaktadır. Arazi boyutunun artması sonucu yeni oluşan noktaların yükseklik değerleri Eşitlik 5.24 kullanılarak hesaplanmaktadır. Bu eşitlikte $f(x,y)$ fonksiyonu x ve y noktasının yükseklik değerini sembolize etmektedir [38].

BÖLÜM 6

SONUÇLAR VE ÖNERİLER

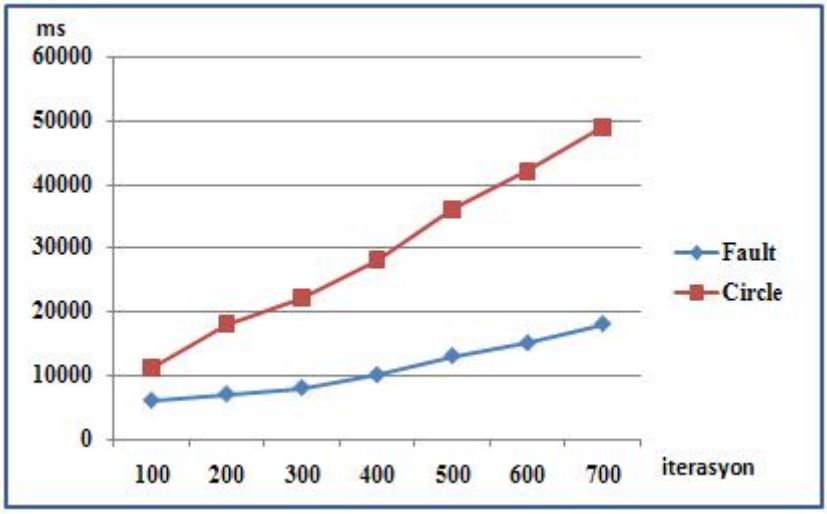
Bu tez çalışmasında 3 boyutlu uygulamalarda kullanılabilecek sanal arazi ortamları oluşturulmuştur. Bu ortamlar hazırlanırken tüm platformlarda kullanılabilmesi için JAVA yazılım dili kullanılmıştır. Böylece platform bağımsız olarak çalışabilen bir sistem gerçekleştirilmiştir.

Sanal arazi ortamları oluşturulurken iki yöntem kullanılmıştır. İlk yöntem ile rastgele arazi ortamları oluşturulmaktadır. Bu yöntem sayesinde herhangi bir veri dosyası kullanılmadan çeşitli tekniklerle araziler otomatik olarak oluşturulmaktadır. Uçuş, tank, araba simülatörleri ve oyunlar gibi sürekli değişen arazi yapısına ihtiyaç duyulan uygulamalar için oldukça kullanışlıdır. Bu yöntemde araziler hızlı bir şekilde oluşturulur. Eğitim simülatörleri gibi uygulamalarda eğitimden geçen bireyin kabiliyetini ve başarı oranını bulabilmek için arazi yapısının değişken bir şekilde olması gerekmektedir. Böylece adayın başarısı test edilebilir.

Rastgele arazi üretme metotları birbirleri ile kıyaslandığında Fault algoritması ile daha gerçekçi arazi modeli oluşturabilmek için iterasyon sayısının diğer algoritmalara oranla daha fazla olması gerekmektedir. Buda daha fazla iş yükü ve işlem kapasitesini gerektirir.

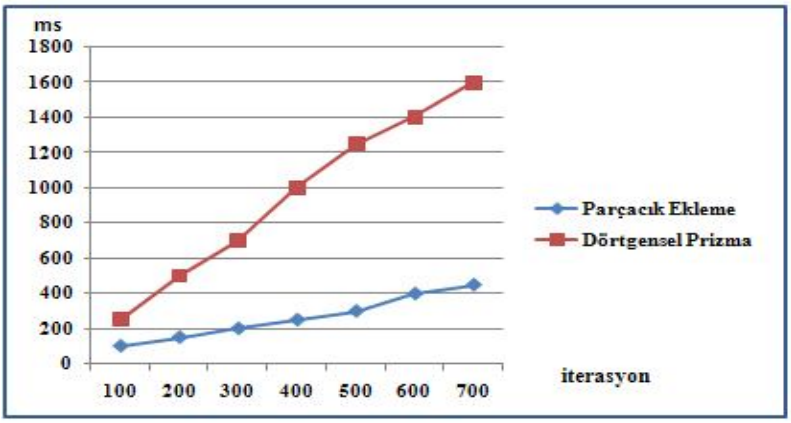
Çember algoritması Fault algoritmasına oranla kullanıcı etkileşimini desteklemektedir. Yeterli seviyede olmamasına karşın çemberin yarıçapının kullanıcıdan alınmasını sağlayarak değişik yapılarda araziler oluşturabilmek mümkündür. Bu algoritmanın dezavantajı; işlem süresinin uzun olmasıdır. İşlem zamanı bakımından Fault algoritması ve Çember algoritması performansı grafiği Şekil 6.1'da gösterilmektedir. İterasyon sayısının artması ile işlem yükünü lineer

olarak artmaktadır. Fakat Çember algoritması Fault algoritmasına göre daha fazla zaman harcamaktadır.



Şekil 6.1. Fault ve Çember algoritması zaman analizi.

Parçacık ekleme algoritması Fault algoritmasına benzer şekilde herhangi bir kullanıcı etkileşimini desteklememektedir. Arazi yapısını oluşturabilmek için iterasyon sayısının oldukça yüksek değerlerde olması gerekmektedir. İterasyon sayısının artması sonucunda elde edilen araziler yüksek ve keskin kenarlı dağlık arazilere benzemektedir. Uçuş simülatörleri gibi keskin manevra kabiliyetini ölçen testlerde bu tip arazilerin kullanılması uygun olacaktır. Bu algoritma bu ihtiyacı oldukça gidermektedir.

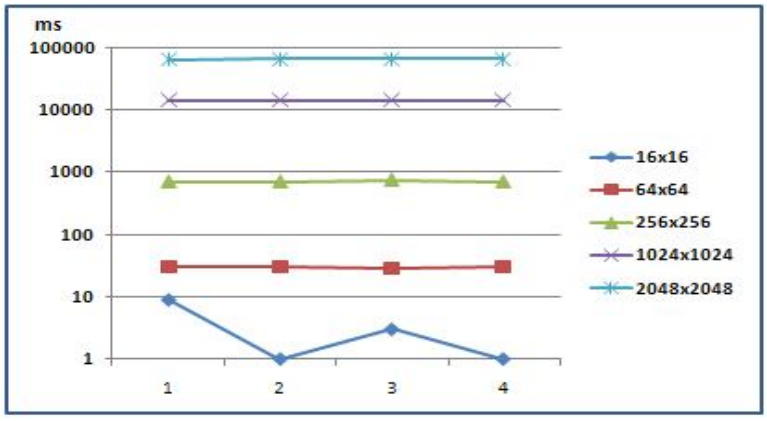


Şekil 6.2. Parçacık ve Dörtgensel prizma ekleme algoritması zaman analizi.

Dörtgensel algoritma parçacık ekleme algoritmasına oldukça benzemektedir. Şekil 6.2'deki grafikte görüldüğü gibi dörtgensel algoritmada iterasyon sayısındaki artışlar zaman faktöründe küçük değişmeleri meydana getirmektedir.

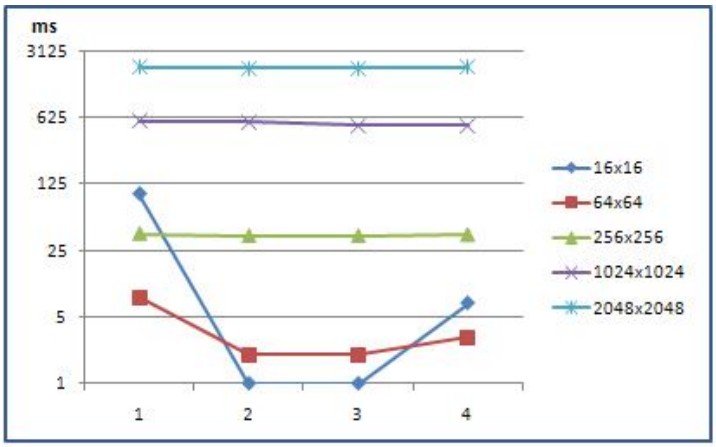
İkinci yöntem ise GIS harita dosyaları kullanılarak gerçek verilerin 3 boyutlu modelinin oluşturulmasıdır. Bu şekilde oluşturulan sanal arazi modeli sayesinde istenilen bir coğrafyanın genel yapısı hakkında fikir sahibi olunabilir. Özellikle askeri uygulamalar açısından önem taşıyan bu dosyalar sayesinde bir ülkenin ya da her hangi bir bölgenin nasıl göründüğü hakkında bir fikir elde edilebilir. Örneğin askeri uçuş eğitimlerinde o ülkenin GIS haritaları kullanılarak pilotun gerçek bir uçuş eğitiminden önce sanal ortamda o arazi yapısı hakkında fikir sahibi olmasını sağlayacaktır. Bu sayede maddi anlamda tasarruf sağlanacaktır. Örneğin gerekli donanım, benzin… gibi gerçek bir uçuş için kullanılabilecek ekipmanlardan tasarruf edilebilecektir. Ayrıca uçuş sırasında daha önce o arazi yapısını görmemiş pilotun deneyim eksikliğinden kaynaklanabilecek hasarlar engellenmiş olacaktır. Pilot adayı sanal ortamda karşılaştığı o bölgenin gerçek ortamdan bir farkının olmadığını anlayabilecek ve beklenmedik olaylarla karşılaşma ihtimali düşecektir. Bu uygulama sadece uçuş simülatörlerinde değil aynı zamanda tank eğitimlerinde kullanılabilmektedir. Böylece askeri harekât öncesinde sanal ortamlarda eğitilen sürücülerin operasyon sırasında yaşanabilecek herhangi bir olumsuzluk ile karşılaşma ihtimali düşmektedir.

Gerçekleştirilen uygulama JAVA tabanlı bir uygulamadır. Bunun için java 1.6 versiyonu ile eclipse ganymede ortamı kullanılmıştır. 3B simülasyon gerçekleştirmek için OPENGL kullanılmıştır. Intel tabanlı Core 2 Quad CPU kullanılmıştır. Şekil 6.3 çeşitli boyutlarda rastgele yükseklik haritası üretimi sırasında geçen toplam zamanı milisaniye cinsinden gösterilmektedir. Ölçümler beşer milisaniye aralıklarla yapılmıştır.



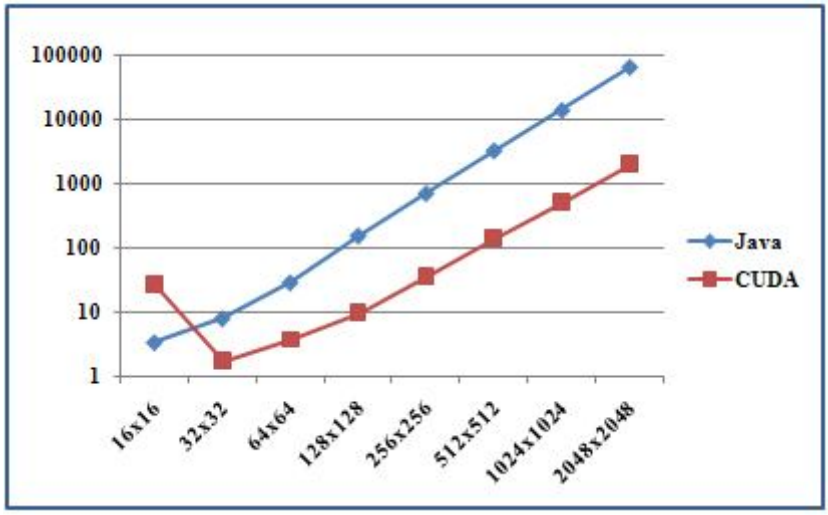
Şekil 6.3. CPU performansı.

Şekil 6.3'de gösterildiği gibi CPU üzerinde yapılan işlemler arttığında toplam işlem süresi artmaktadır. *16x16* boyutlarında bir arazi modeli üretimi için geçen toplam süre ortalama olarak 8 ms iken *256x256* boyutlarındaki bir arazi modelini oluşturmak için geçen toplam süre ortalama 716 ms'dir. Arazi boyutunun artmasına bağlı olarak işlem süresinin artması uygulamayı olumsuz etkilemektedir. İşlem yükünü azaltmak mümkün olmadığından toplam zamanı azaltmak için NVIDIA CUDA teknolojisi kullanılmıştır. Bu teknoloji sayesinde yapılacak olan işlemler GPU ile paralel bir biçimde yürütülmüştür. Ekran kartı olarak NVIDIA GeForce 9500 GT kullanılmıştır. Uygulama Java tabanlı bir uygulama olduğu için CUDA' nın JAVA ile birlikte kullanılabilmesi için geliştirilmiş olan JCUDA kütüphanesi kullanılmıştır. CUDA içerisinde çeşitli kütüphaneler bulundurmaktadır. Bu kütüphaneler içerisinden uygulamamıza uygun olan CUDA CUFFT Kütüphanesi kullanılmıştır.



Şekil 6.4. GPU Performans grafiği.

Şekil 6.4. beşer milisaniye aralıklarla alınmış test sonuçlarını göstermektedir. Grafikte görüldüğü gibi CUDA küçük boyutlu uygulamalarda performans artışı sağlayamamaktadır. *16x16* boyutlarında bir yükseklik haritası için geçen toplam süre ortalama 26 milisaniye iken *256x256* boyutlarındaki bir yükseklik haritasının oluşturulması için geçen toplam süre ortalama 35 milisaniyedir. Küçük boyutlardaki uygulamalarda geçen zaman CPU ile kıyaslandığında oldukça yetersiz görünmektedir. Bu durumlarda CPU kullanılması daha etkili olmaktadır. Fakat oluşturulan yükseklik haritasının boyutu büyüdükçe CUDA'nın sağlamış olduğu performans artmaktadır.



Şekil 6.5. CPU GPU Performans grafiği.

Şekil 6.5 CPU ve GPU kullanımı sonucu ölçülen sürelerin milisaniye cinsinden karşılaştırılması verilmektedir. Küçük boyutlu uygulamalarda CPU'dan daha kötü performans sağlayan CUDA büyük boyutlu uygulamalarda oldukça yüksek bir performans sağlamaktadır. *2048x2048* boyutlarındaki bir yükseklik haritasının oluşturulması için geçen toplam süre ortalama olarak CUDA ve GPU ile 2051 ms iken Java ve CPU ile yapıldığında ortalama 65000 milisaniye sürmektedir. CUDA yüksek boyutlarda ki uygulamalarda 30x performans sağlayabilmektedir.

KAYNAKLAR

1. Kamal, K. R. and Udin, Y. S., "Parametrically controlled terrain generation" ***Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia***, 2 (2): 111-119 (2007).

2. Lechner, T., Ren, P., Watson, B., Brozefski, C., and Wilenski, U., "Procedural modeling of urban land use**".** ***In International Conference on Computer Graphics and Interactive Techniques***, 135-138 (2006).

3. Li, Q., Wang, Q., Zhou, F., Tang X. and Yang, K., "Example-based realistic terrain generation", ***Lecture Notes in Computer Science***, 4282: 811 (2006).

4. Brosz, J., Samavati, F.F. and Sousa, M.C., "Terrain synthesis by- example" Proc. ***First Int'l Conf. Computer Graphics Theory and Applications*****,** 58-77 (2006) **.**

5. Göktaş, H. H., Çavuşoğlu, A., Şen, B., "AUTOCITY: A System for Generating 3D Virtual Cities for Simulation Systems on GIS Map", ***AutoSoft - Intelligent Automation and Soft Computing***, USA, 15 (1): 29-39, (2009).

6. Doran, J. and Parberryy, I., "Controlled procedural terrain generation using software agents" ***Computational Intelligence and AI in Games, IEEE Transactions on***, 111 – 119 (2010).

7. Üstüntaş, T., "Sayısal arazi modellerinde hassasiyet analizi ve enterpolasyon yöntemleri", Yüksek Lisans Tezi, ***Selçuk Üniversitesi Fen Bilimleri Enstitüsü***, Konya ,25-40 (1994).

8. Ackerman, F., "Techniques and strategies for DEM generation, digital photogrammetry", ***An Addendum to the Manual of Photogrammetry,*** 6: 135, USA. (1996).

9. Özer, H., "Sayısal Arazi Modeli Oluşturma Yöntemleri ", ***Harita Genel Komutanlığı Dergisi*****,** Ankara, 102: 15 (1989).

10. Öztürk, E. ve Koçak, E., "Farklı kaynaklardan değişik yöntem ve ölçeklerde üretilen sayısal yükseklik modellerinin doğruluk araştırması", ***Harita Genel Komutanlığı Dergisi***, 137, (2006).

11. Demirci, F., "Filyos havzasındaki sediment birikim alanlarının uydu görüntü verileri ve sayısal arazi modeli ile analizi", Yüksek Lisans Tezi, ***İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 24-35 (2008).

12. Pınarcı, E., "İki boyutlu kalman filtresinin yersel lazer tarama verisine uygulanması", Yüksek Lisans Tezi, ***Gebze İleri Teknoloji Enstitüsü Mühendislik ve Fen Bilimleri Enstitüsü,*** Gebze, 4-9 (2007).

13. Canıberk, M., "GBS ve lazer destekli bir CBS veri toplama sisteminin yersel fotogrametri ile bütünleştirilmesi", Yüksek Lisans Tezi, ***Selçuk Üniversitesi Fen Bilimleri Enstitüsü***, Konya, 17-48 (2008).

14. Köse,M. H., "Uydu radar görüntülerinden üç boyutlu sayısal arazi modelinin üretilmesi", Yüksek Lisans Tezi, ***Selçuk Üniversitesi Fen Bilimleri Enstitüsü***, Konya, 15-40 (2006).

15. Sefercik; U. G., "Radar interferometri tekniği İle sym üretimi ve doğruluk değerlendirmeleri", ***TMMOB Harita ve Kadastro Mühendisleri Odası 11.Türkiye Harita Bilimsel ve Teknik Kurultayı***, 2-6 (2007).

16. İnternet: TÜBİTAK "Uzay" **http://www.uzay.tubitak.gov.tr/** (10/10/201).

17. Karaş, İ. ve Eroğlu, S., "Demir çelik endüstrisinde coğrafi bilgi sistemlerinin kullanımı", ***Kuruluşundan Bugüne Karabük ve Demir Çelik Sempozyumu***, Karabük, (2010).

18. İnternet: USGS "DEM" **http://rockyweb.cr.usgs.gov** (25/04/2011).

19. Farr, T.G. and Kobrick, M. "Shuttle radar topography mission produces a wealth of data", ***Amer Geophys Union Eos***, 81: 583-585 (2000).

20. İnternet: USGS "DTED" **http://eros.usgs.gov** (2010).

21. İnternet: National Imagery and Mapping Agency (NIMA), "Digital Terrain Elevation Data " **http://egsc.usgs.gov/nimamaps/** (2000).

22. Kekeç, B., "Effects of parallel programmıng design Patterns on the performance of multi-core processor based real time embedded systems", ***A Thesıs Submıtted To The Graduate School Of Natural And Applıed Scıences Of Mıddle East Technıcal Unıversıty*** , Ankara, 4-12 (2010).

23. İnternet: Lighthouse "OpenGL" **http://www.lighthouse3d.com/** (09.12.2010).

24. İnternet: Waterloo Üniversitesi "PMath 370 Chaos and Fractals" **http://www.math.uwaterloo.ca/** (02.02.2011).

25. Şahin, H. ve Külür, S., "Gpgpu yöntemi ile fotogrametrik uygulamalar", ***TUFUAB 2011, VI. Teknik Sempozyumu***, Antalya (2011).

26. Owens, J.D. Houston, M. Luebke, D. Green, S. Stone, J.E. and Phillips, J.C., "GPU computing ", ***Proceedings of the IEEE***, 96(5): 879 – 899 (2008).

27. İnternet: Top500 **"**Superbilgisayarlar Listesi" **http://www.top500.org/** (10.04.2011).

28. Vasiliadis, G., Antonatos, S., Polychronakis, M., Markatos E. P., and Ioannidis, S., "Gnort: High Performance Network Intrusion Detection Using Graphics Processors", ***Proceedings. 2004 International Symposium***, Boston, MA, USA 116-134 (2008).

29. Chatz, M.C., Trapnell, C., Delcher, A.L. and Varshney, A., "High-throughput sequence alignment using Graphics Processing Units.", ***BMC Bioinformatics***, 8 (1): 474 (2007).

30. Manavski, S. A. and Valle, G., "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment". ***BMC Bioinformatics,*** 9 (2): 10 (2008).

31. İnternet: Pyrit "Google Code" **http://code.google.com/p/pyrit/** (01.05.2011).

32. Hussin, F. A., Nazlee A. M. and Ali, N. B. Z. , "Tranformation of CPU-based applications to leverage on graphics processors using CUDA", ***International Journal of Electrical & Computer Sciences IJECS-IJENS,*** 10 (1): 40 - 47 (2003).

33. İnternet: NVIDIA "CUDA C programming guide" **http://www.nvidia.com/,** (11/09/2010).

34. İnternet: ATI "AMD " **http://www.amd.com/us/products/technologies/streamtechnology** (2010).

35. İnternet: Pavia Üniversitesi "Mr. Alessandro Martinelli ders notları" **http://ingegneria.unipv.it/didattica/schedacorso1011.php?cod=064132&spec=0** (10.10.2010).

36. Bradley, C. J.,"The algebra of geometry: cartesian, areal and projective co-ordinates", ***Highperception***, Bath, 100-300 (2007).

37. İnternet: M.A. Yükselen, "İstanbul teknik üniversitesi uygulamalı sayısal yöntemler ders notları " **http://web.itu.edu.tr/~yukselen/HM504/** (2011).

38. Liu, J.J., Zhang, H. and Chen, L., "Bilinear interpolation of geomagnetic field" ***Computer Application and System Modeling (ICCASM)***, 2: 665-668 (2010).

ÖZGEÇMİŞ

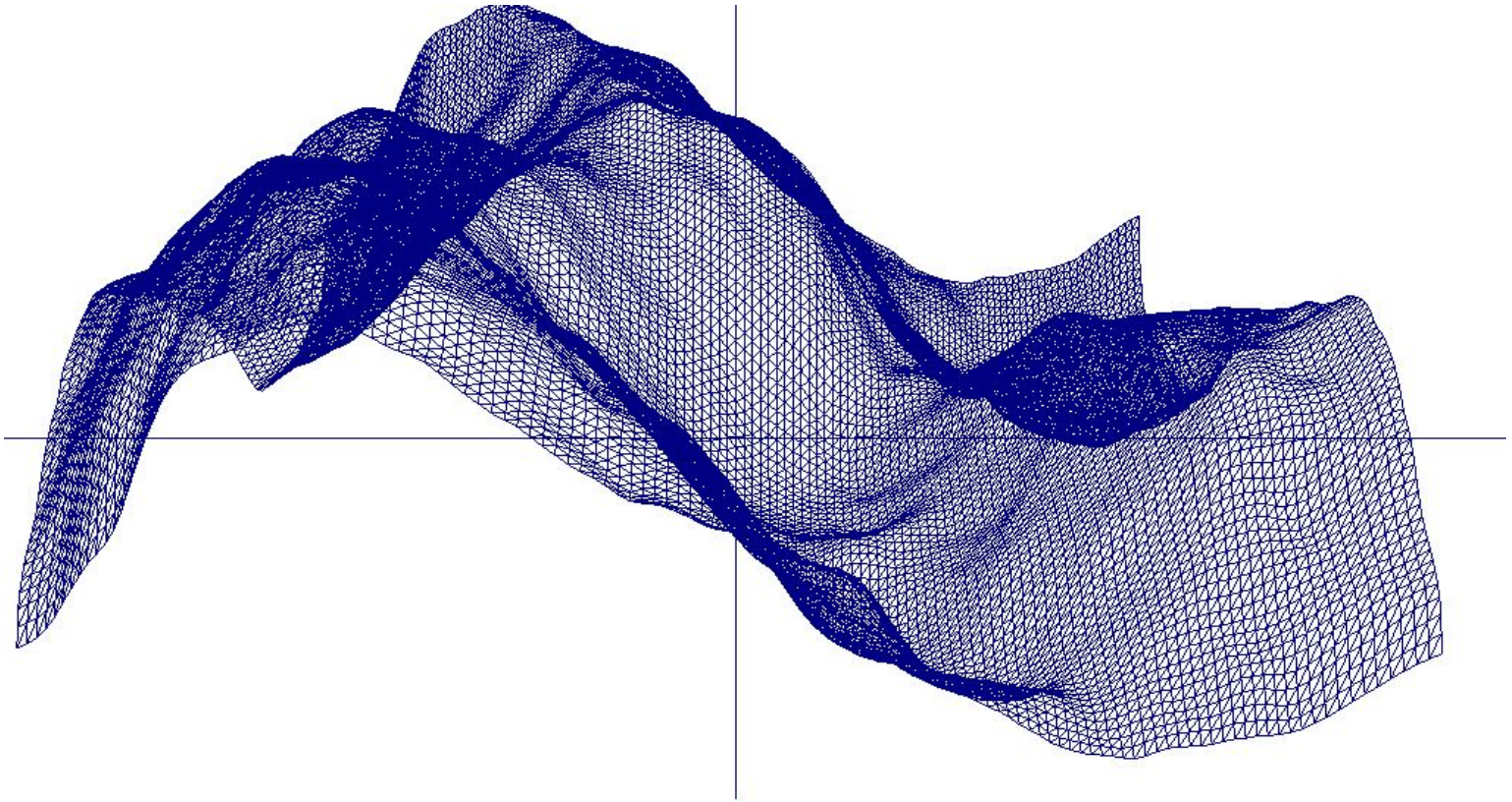
Serpil EROĞLU 1987 yılında Diyarbakır'da doğdu; ilköğretim ve lise öğrenimini aynı şehirde tamamladı. Nevzat Ayaz Anadolu lisesinde mezun oldu. 2005 yılında Harran Üniversitesi Bilgisayar Mühendisliği Bölümünde üniversite eğitimine başladı. 2009 yılında Karabük Üniversitesinde Bilgisayar Mühendisliği bölümü Yazılım Anabilim Dalında araştırma görevlisi olarak göreve başladı. 2009 yılında Karabük Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda başlamış olduğu yüksek lisans programını 2011 yılında tamamladı. Halen KBÜ Mühendislik Fakültesi Bilgisayar Mühendisliği programında Araştırma Görevlisi olarak görev yapmaktadır.

ADRES BİLGİLERİ

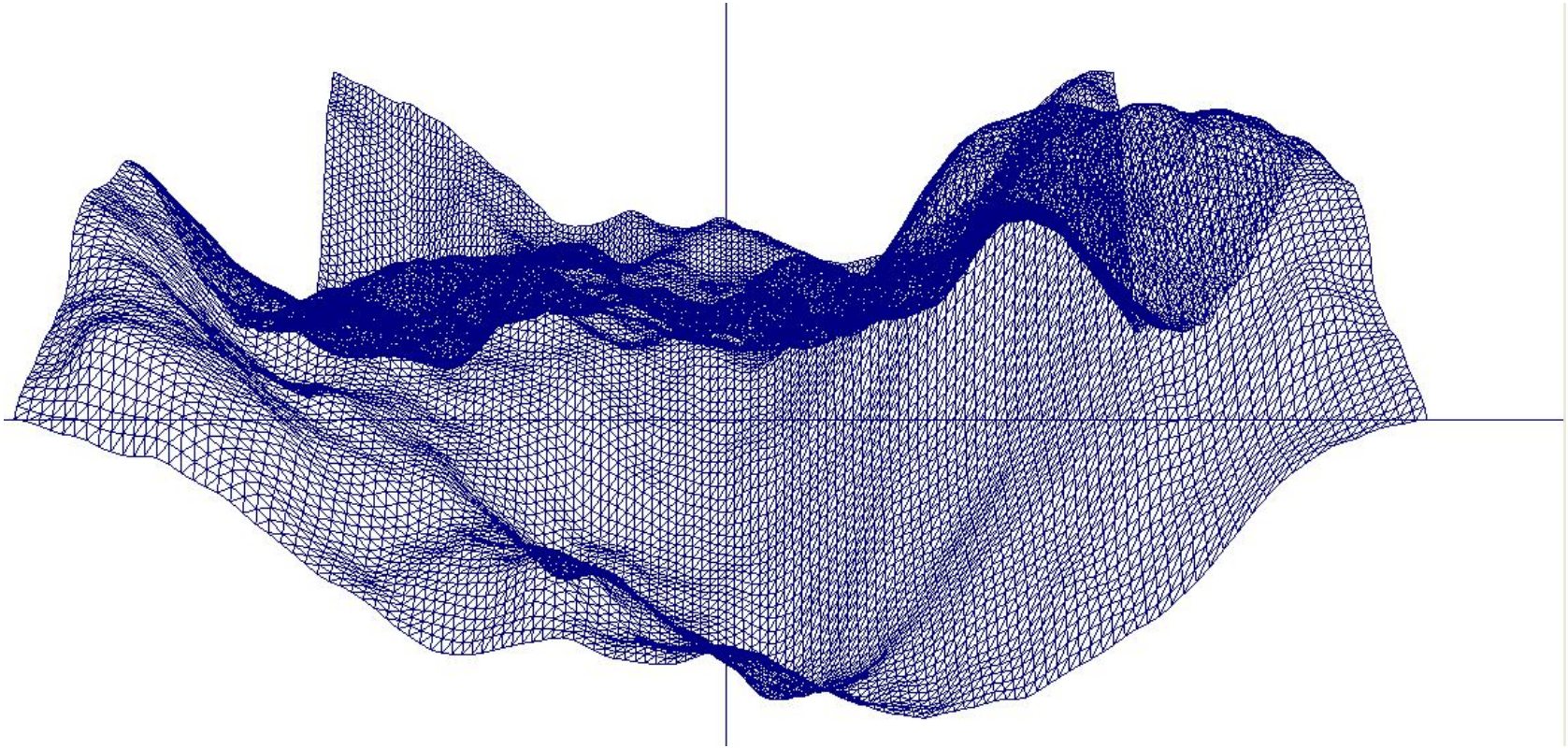
Adres : Karabük Üniversitesi
Mühendislik Fakültesi
Balıklar kayası Mevkii / KARABÜK

Tel : 0 370 433 20 21
E-posta : seroglu@karabuk.edu.tr

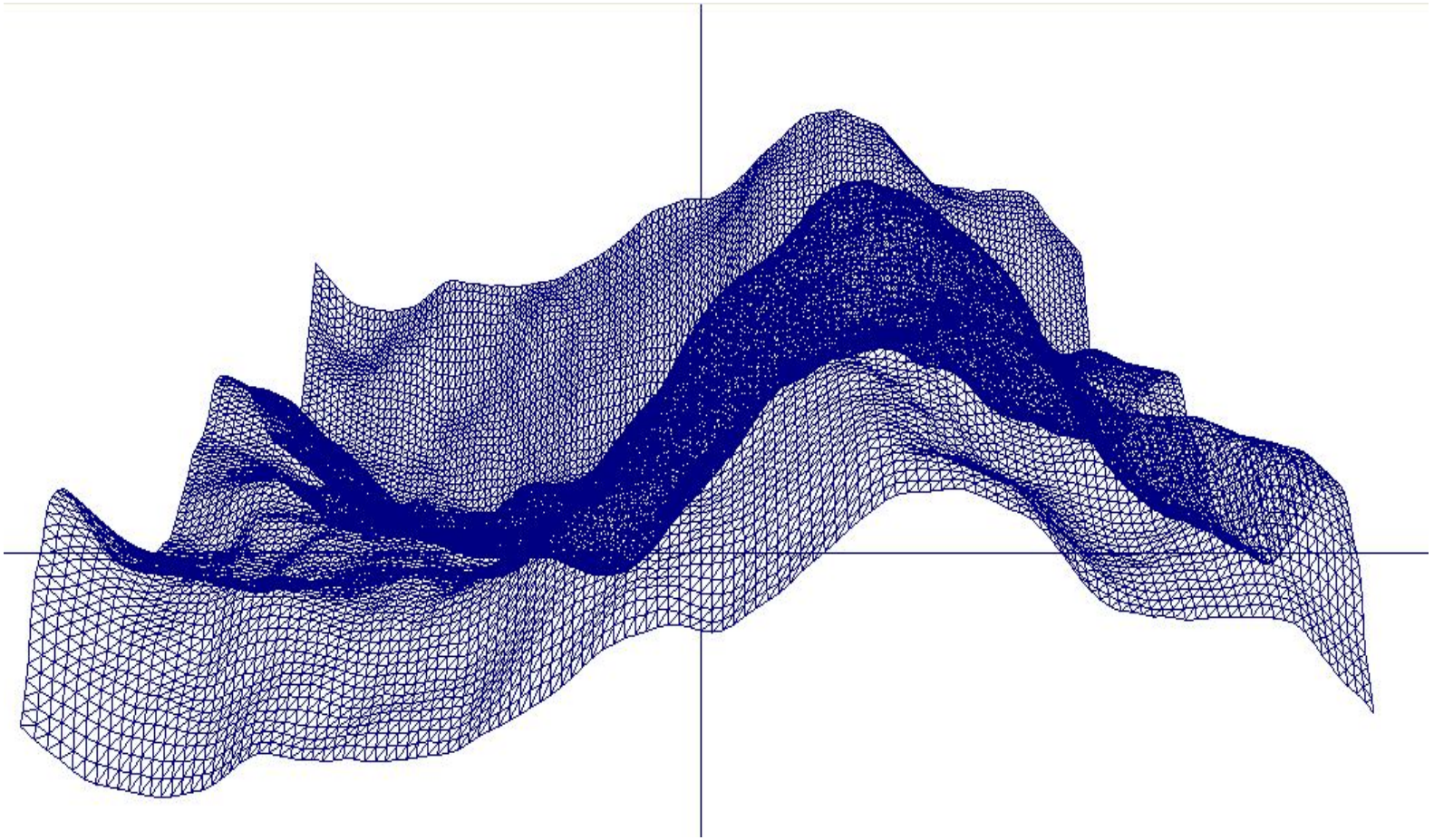
EK AÇIKLAMALAR A.
RASTGELE ALGORİTMALAR SONUCU OLUŞTURULMUŞ ARAZİ MODELLERİ



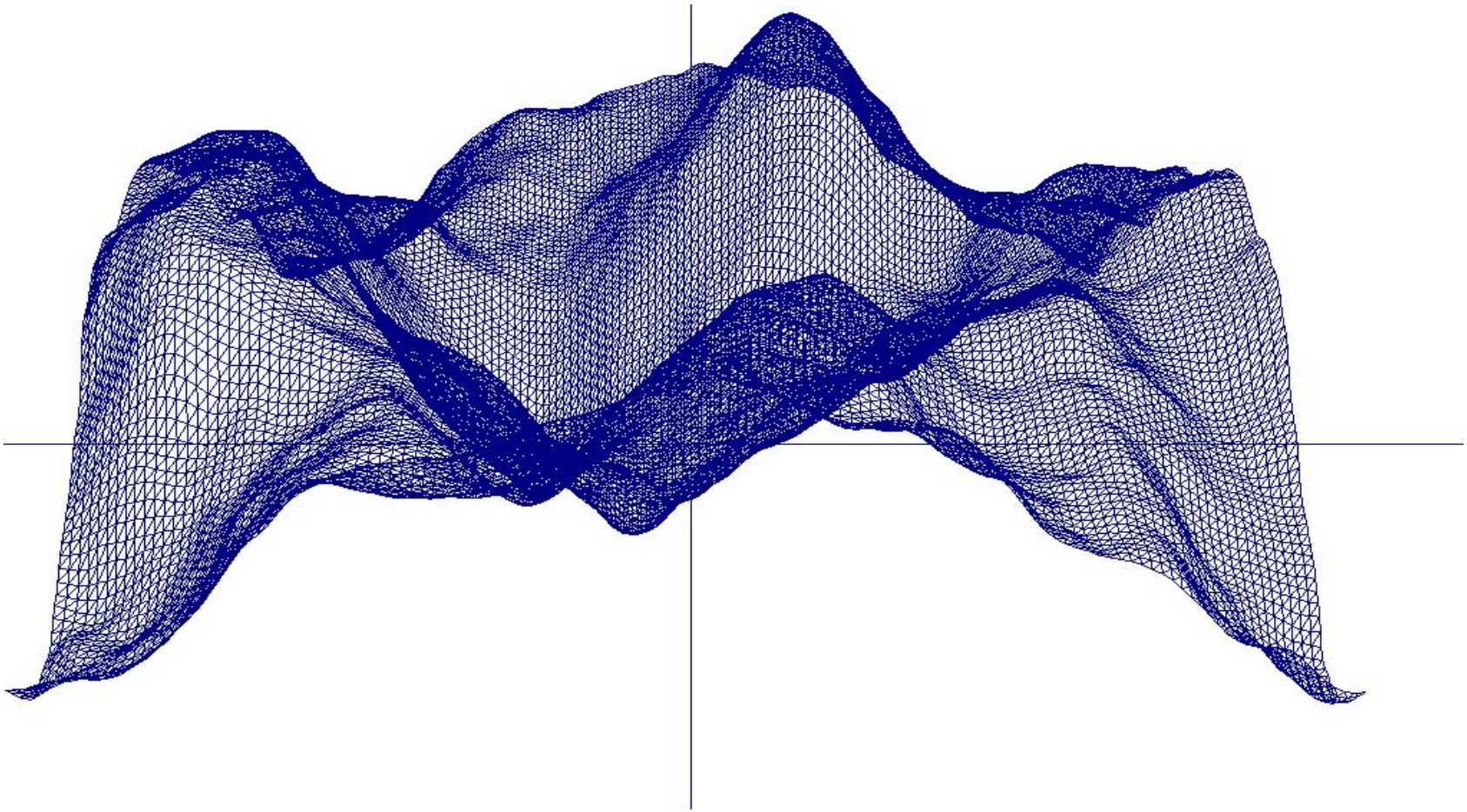
Şekil Ek A.1. FFT Algortiması sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.



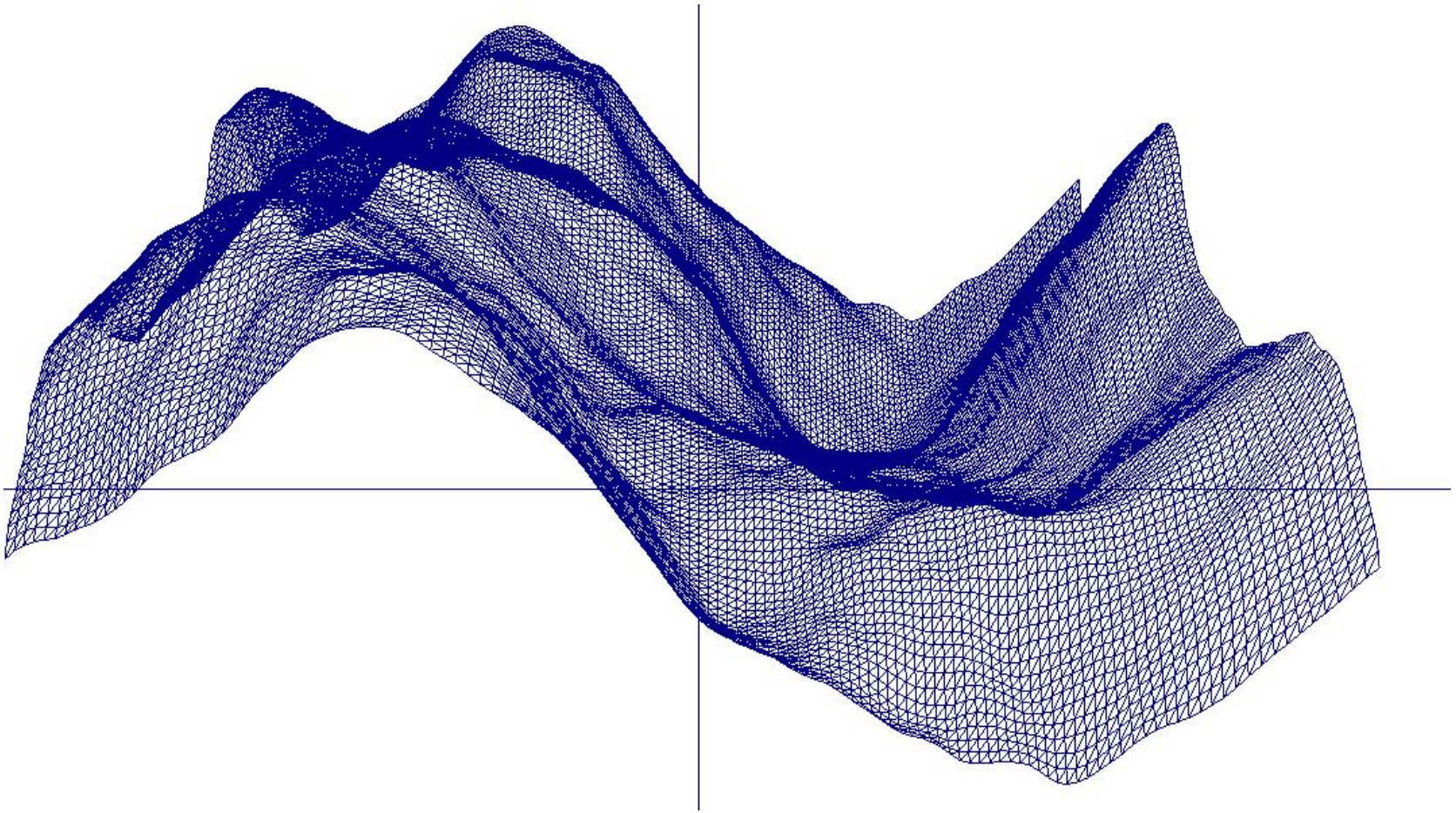
Şekil Ek A.2. FFT Algortiması sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.



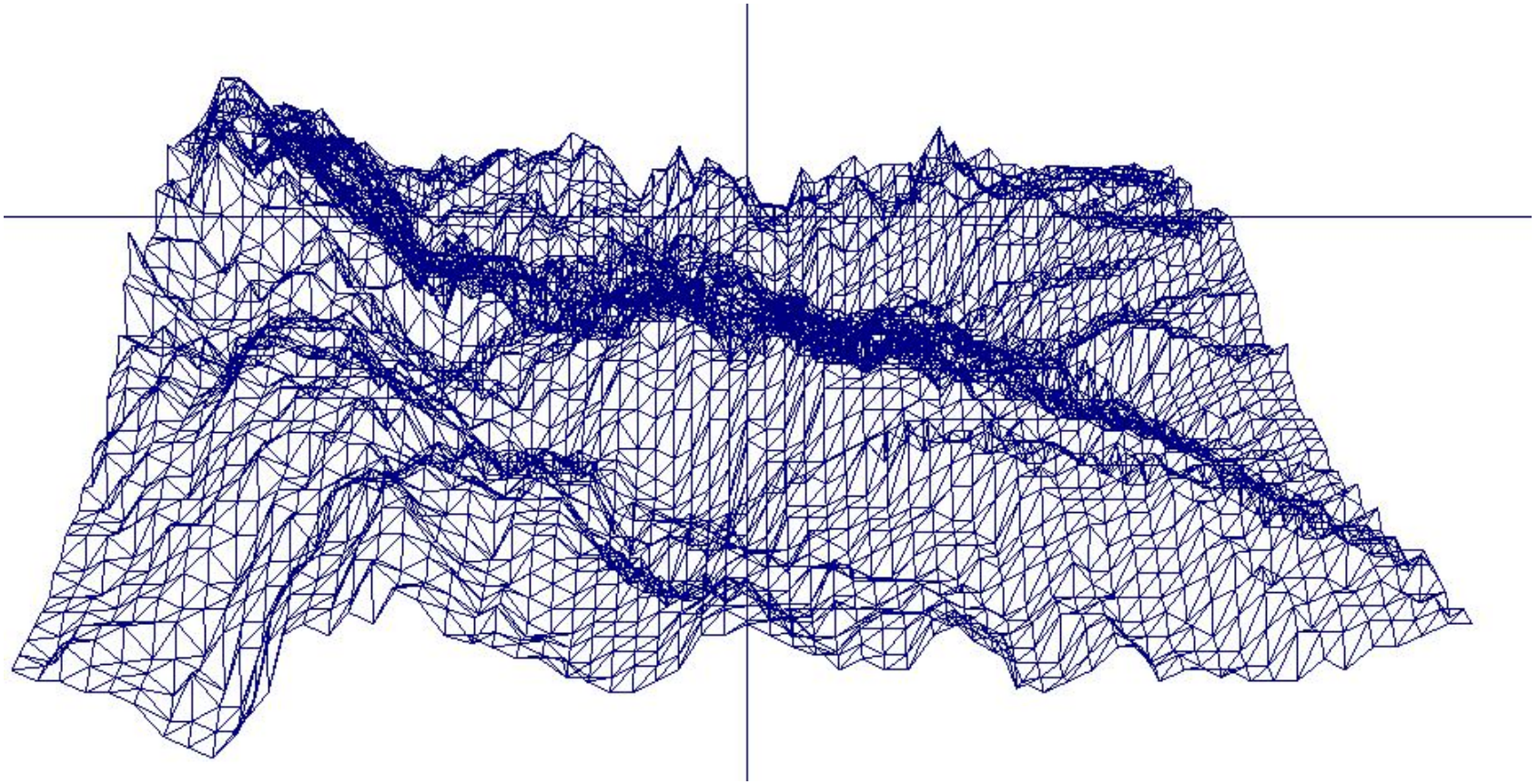
Şekil Ek A.3. FFT Algoritması sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.



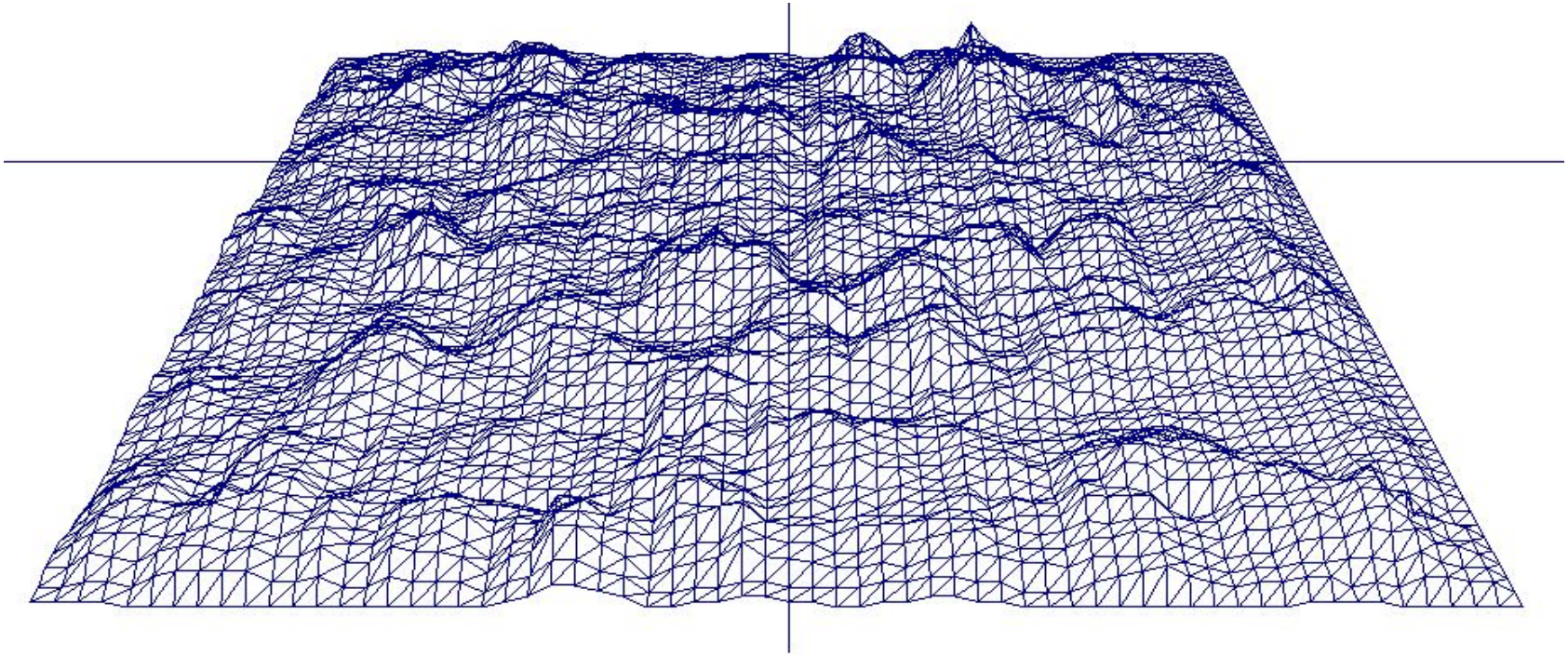
Şekil Ek A.4. FFT Algoritması sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.



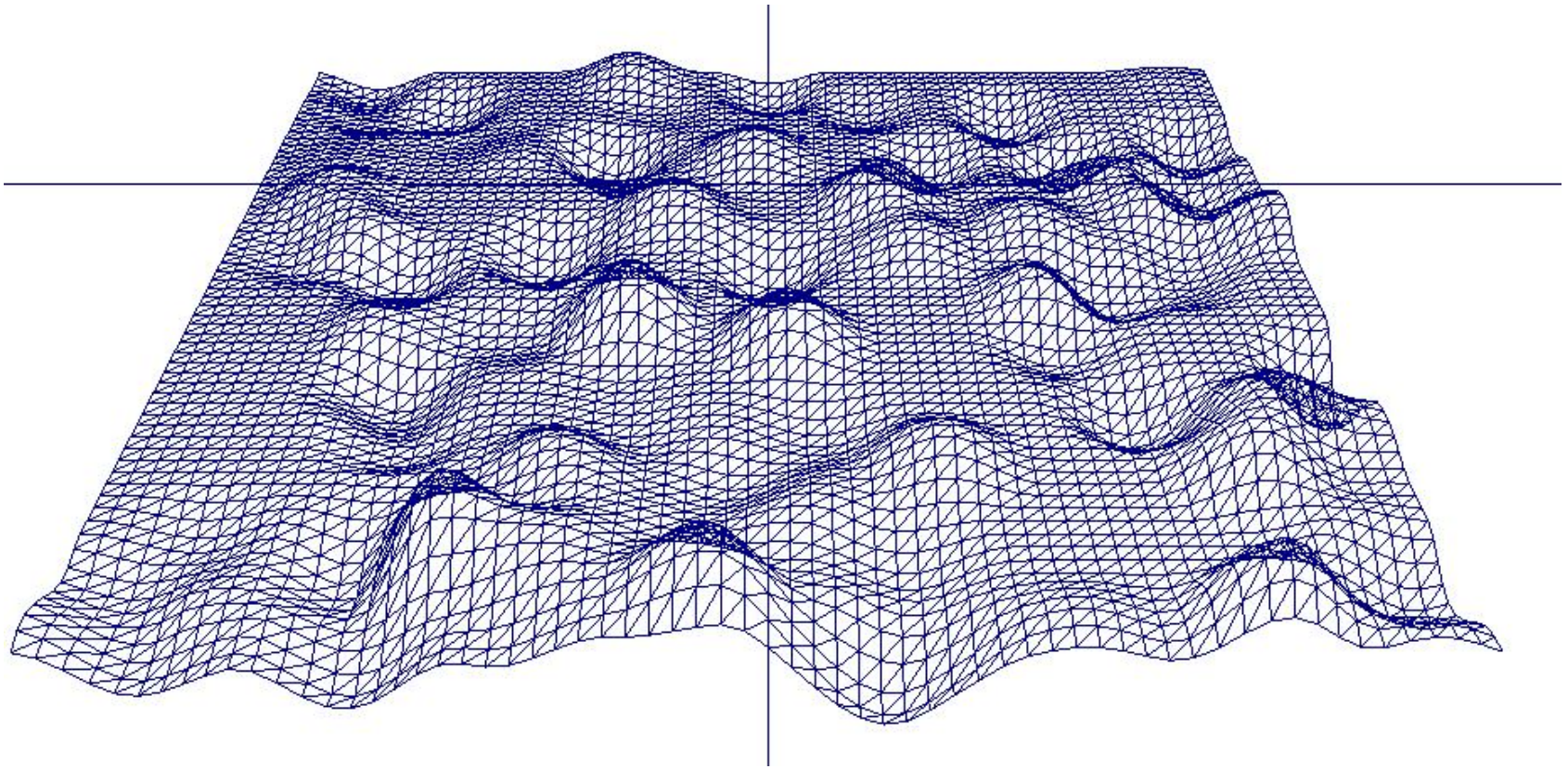
Şekil Ek A.5. FFT Algoritması sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.



Şekil Ek A.6. Fault Algortiması ile 500 iterasyon sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.



Şekil Ek A.7. Dörtgensel Algortiması ile 1000 iterasyon sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.



Şekil Ek A.8. Çember Algoritması ile 500 iterasyon sonucu oluşturulmuş $128x128$ boyutlarındaki arazi modeli.

EK AÇIKLAMALAR B
PROGRAM KODLARI

```
/*
     * FAULT Algoritması
     *
     * */

public void createFaultTerrain(int numIterations, float[][] t)

{
      float dispAux;
      int i,j,k;
      float a,b,c,w,d;

      int n=size;

      for (k = 0; k < numIterations;k++) {


         d = (float) Math.sqrt(n * n + n * n);
         w = randomGenerator.nextInt(n*n);
         a = (float) Math.sin(w);
         b = (float) Math.cos(w);
         c = (float) randomGenerator.nextFloat()*(d-2)-(d-2)/2;


         for (i = 0;i < n; i++)
               for(j = 0; j < n; j++)
         {
                  if (i * a + j * b + c > 0)
                     dispAux = maxDisp;
                  else
                     dispAux = -maxDisp;

                     t[i][j] += dispAux;
            }
      }

     }//Fault sonu


    /**
     * ÇEMBER ALGORİTMASI
     *
     * */


 public void createCircleTerrain(int numIterations,int size,float[][] t)
      {

      float dispAux;
      int i,j,k,dispSign;
      float x,z,r,pd;
      int n=size;
       int iterationsDone = 0;
       float  minDisp=0.1f,disp;
```

```
        int itMinDisp=100;
        float terrainCircleSize  = 100.0f;


       for (k = 0; k < numIterations;k++) {

           z = (randomGenerator.nextFloat()) * n;
           x = (randomGenerator.nextFloat()) * n;

           iterationsDone++;

           if (iterationsDone < itMinDisp)
               disp = (float) (maxDisp + (iterationsDone/(itMinDisp+0.0))* (minDisp -
maxDisp));

   else
               disp = minDisp;
           r = (randomGenerator.nextFloat());
           if (r > 0.5)
               dispSign = 1;
   else
               dispSign = -1;

           for (i = 0;i < n; i++)
               for(j = 0; j < n; j++) {
                       pd = (float) (Math.sqrt(((i-x)*(i-x) + (j-z)*(j-z)) / terrainCircleSize)*2);

                       if (pd > 1) dispAux = 0.0f;

                       else if (pd < -1) dispAux = 0.0f;

                       else
                           dispAux =  (float) (disp/2*dispSign + Math.cos(pd*3.14)*disp/2 *
dispSign);

                   t[i][j] += dispAux;
               }
        }


       }



    /**
      * DÖRTGENSEL ALGORİTMA
      *
      * */

     public void createTerrainQuad(int numIterations,float[][] t)
      {

      int cevre=2;
      System.out.println("Quad için :"+numIterations);
```

```
int x1,z1,yon;
int n=size;


for(int i=0;i<numIterations;i++)
{
    x1= randomGenerator.nextInt(size);
    z1= randomGenerator.nextInt(size);
    yon= randomGenerator.nextInt(2);


    if((x1-cevre)>=0 && (x1+cevre<n) &&( z1-cevre)>=0 && (z1+cevre<n) )
    {
        if(yon==1)              {
            for(int j=x1-cevre;j<=x1+cevre;j++)
            {
                for(int k=z1-cevre;k<z1+cevre;k++)
                {
                    t[j][k]+=maxDisp;

                }
            }
            t[x1][z1]+=maxDisp-0.003f;
        }//yon if
    }

}


}


/**
 * PARÇACIK EKLEME
 *
 * */

public void createParticleTerrain(int numIt, int size,float[][] t)
{
 int x,z,i,dir;

 int n=size;
  x = randomGenerator.nextInt(n) ;
 z = randomGenerator.nextInt(n) ;

 for (i=0; i < numIt; i++) {


    dir = randomGenerator.nextInt(4) ;


    if (dir == 2) {
```

```
            x++;
            if (x >= n)
                x = 0;
        }
        else if (dir == 3){
            x--;
            if (x == -1)
                x = n-1;
        }

        else if (dir == 1) {
            z++;
            if (z >= n)
                z = 0;
        }
        else if (dir == 0){
            z--;
            if (z == -1)
                z = n - 1;
        }
             t[x][z] += maxDisp;

    }
     }



/**
 * FFT Algoritması
 *
 * */

 private double rnd (Random r) {
 return 2. * r.nextDouble () - 1.0;
}


public void createFFT(int size)
{
  float[][] t= new float[size][size];
    Random random= new Random();
     // Random arazi değerleri
     for (int i = 0; i < size; ++i) {
       for (int j = 0; j < size; ++j) {
            t[i][j] = (float) rnd(random);
            }
     }
     System.out.println(size+"  Random Yükseklik  değerleri oluştu");

     if(rdiJava.isSelected())
     {
        System.out.println("Java Seçilmiş");
             Timer.createTimer("Java Timer");
             Timer.startTimer("Java Timer");
```

```
                Complex[][] freq = Fourier.FFT2(t);
                 for (int i = 0; i < size; ++i) {
                  for (int j = 0; j < size; ++j) {
                      if (0 == i && 0 == j) continue;
                      double x = Math.min(i, size - i);
                      double y = Math.min(j, size - j);// * 2.0;
                      /double scale = Math.pow(1/Math.sqrt(x*x + y*y), r);
                      freq[i][j] = freq[i][j].multiply(scale);

                  }
                 }
                 t = Fourier.IFFT2(freq);
                 Timer.stopTimer("Java Timer");
                 Timer.prettyPrint();
             }
            else if(rdiCuda.isSelected())
            {
                Timer.createTimer("JCuda Timer");
                Timer.startTimer("JCuda Timer");

                float data[] = make1D(size,t);

                 cufftHandle plan = new cufftHandle();
                cuffPlan2d(plan, size, size, CUFFT_C2C);
                cuffExecC2C(plan, data, data, CUFFT_FORWARD);

                for (int i = 0; i < size; i++) {
                  for (int j = 0; j < size; j++) {
                      if (0 == i && 0 == j) continue;
                      double x = Math.min(i, size - i);
                      double y = Math.min(j, size - j);// * 2.0;
                      double scale = Math.pow(1/Math.sqrt(x*x + y*y), r);
                      data[i*size*2+j*2]=(float) (data[i*size*2+j*2]*scale);
                      data[i*size*2+j*2+1]=(float) (data[i*size*2+j*2+1]*scale);
                  }
                 }
              cuffExecC2C(plan, data, data, JCufft.CUFFT_INVERSE);
              for (int i = 0; i < size; i++)
              {
               data[i]/=data.length;
              }

              t=make2D(size, data);
              Timer.stopTimer("JCuda Timer");
               Timer.prettyPrint();
            }
            else
                JOptionPane.showMessageDialog(terrainOptionsPanel, "Performans modunu
seçiniz ");

            min = max = t[0][0];
             for (int i = 0; i < size; ++ i)
```

```
        for (int j = 0; j < size; ++ j)
          if (t[i][j] < min) min = t[i][j];
          else if (t[i][j] > max) max = t[i][j];

      float alt;
       for (int i = 0; i < size;  i++)
       { for (int j = 0; j < size;  j++)
             {
                alt = t[i][j ];
       t[i][j]=(alt - min) / (max - min);
       //System.out.println(t[i][j]);
             }
      }

    GLRenderer.localterrain=t;


}//Fonksiyon sonu

public float[] make1D(int size,float[][] array)
{
    float[] data = new float[size*size*2];
    int index=0;
    for (int i = 0; i < size;  i++)
    {
          for (int j = 0; j < size; j++)
          {
           data[index]=array[i][j];
           data[index+1]=0;
           index+=2;
         }
    }

    return data;
}
public float[][] make2D(int size,float[] array)
{
    float[][] data = new float[size][size];
    int index=0;
    for (int i = 0; i < size;  i++)
    {
          for (int j = 0; j < size; j++)
          {
           data[i][j]=array[index];
           index+=2;
            }
    }
    return data;
}

private void import_imageActionPerformed(ActionEvent evt) {
    float[][] terrain = null;
   importimage=true;
```

```
String file="";
System.out.println("image imported : ");

chooser = new JFileChooser();
chooser.setCurrentDirectory(new java.io.File("."));
chooser.setDialogTitle("");
chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
//
// disable the "All files" option.
//
chooser.setAcceptAllFileFilterUsed(false);
//
if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {


  file=""+chooser.getSelectedFile();

  }
else {
  System.out.println("No File Selection ");
  }

try {
        BufferedImage image=ImageIO.read(new File(file));
        int terrainWidth=image.getWidth();
        int terrainHeight=image.getHeight();

    terrain=new float[terrainWidth][terrainWidth];
        System.out.println("width : "+terrainWidth);

        for(int i=0;i<terrainHeight;i++){
            for(int j=0;j<terrainWidth;j++){
                int color=image.getRGB(j, i);

                terrain[i][j]=((color & 0xff)*0.01f)/5;
                                }
        }


} catch (IOException e) {
        e.printStackTrace();
   }

GLRenderer.localterrain=terrain;

}//GEN-LAST:event_aboutButtonActionPerformed


private void import_Dted(ActionEvent evt) {

   System.out.println("Start Decoding");

  byte[] fileArray = null;
  float[][] terrain=null;
```

```
        String filename="";
          System.out.println("Dted imported : ");

         chooser = new JFileChooser();
         chooser.setCurrentDirectory(new java.io.File("."));
         chooser.setDialogTitle("");
         chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
         //
         // disable the "All files" option.
         //
         chooser.setAcceptAllFileFilterUsed(false);
         FileFilter myfilter = new ExtensionFileFilter("DT0 and DT1", new String[] { "DT0",
"DT1" });
         chooser.setFileFilter(myfilter);
         //
         if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {


          filename=""+chooser.getSelectedFile();

          }
         else {
          System.out.println("No File Selection ");
          }

        try {
         fileArray = getBytesFromFile(new File(filename));
          String longitude="";

         for ( int i = 47; i < 51; i++)
         {
          longitude+=(char)fileArray[i];
         }

         System.out.println("number of longitude "+longitude);
         String latitudue="";

         for ( int i = 51; i < 55; i++)
         {

            latitudue+=(char)fileArray[i];

         }


         System.out.println("number of latidue "+latitudue);

          terrain= new float[Integer.parseInt(longitude)][Integer.parseInt(latitudue)];

          int step=3428;

          for(int i=0;i<Integer.parseInt(longitude);i++)
          {
```

```
        int j=0;
        step=step+8;

         while(j<Integer.parseInt(latitudue))
         {

           int y=arr2int(fileArray,step);
            step+=2;
            terrain[i][j]=y*0.0001f;
           j=j+1;
         }
          step=step+4;

      }
  } catch (IOException e) {
      e.printStackTrace();
   }
     GLRenderer.localterrain=terrain;

   }//GEN-LAST:import_Dted


private void import_Srtm(ActionEvent evt) {

     System.out.println("Start Decoding");

       byte[] fileArray = null;
       float[][] terrain=null;
       String filename="";
         System.out.println("Dted imported : ");

       chooser = new JFileChooser();
       chooser.setCurrentDirectory(new java.io.File("."));
       chooser.setDialogTitle("");
       chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
       //
       // disable the "All files" option.
       //
       chooser.setAcceptAllFileFilterUsed(false);
       FileFilter myfilter = new ExtensionFileFilter("HGT", new String[] { "HGT" });
       chooser.setFileFilter(myfilter);
       //
       if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {


         filename=""+chooser.getSelectedFile();

         }
       else {
         System.out.println("No File Selection ");
         }
      try {
         fileArray = getBytesFromFile(new File(filename));
         System.out.println(fileArray.length);
```

```
        int row=1201;
        int col=1201;
       terrain= new float[row][col];
        int index=0;

        for(int i=0;i<row;i++)
        {
          //int index=0;
          for(int j=0;j<col;j++)
          {
            int value =arr2int(fileArray,index);
            terrain[i][j]=value*0.0001f;
            index+=2;

          }

        }

        System.out.println(index);
      } catch (IOException e) {
        e.printStackTrace();
      }

    GLRenderer.localterrain=terrain;

}

private static byte[] getBytesFromFile(File file) throws IOException {

    InputStream is = new FileInputStream(file);
    System.out.println("\nDEBUG: FileInputStream is " + file);

    // Get the size of the file
    long length = file.length();
    System.out.println("DEBUG: Length of " + file + " is " + length + "\n");

    if (length > Integer.MAX_VALUE) {
      System.out.println("File is too large to process");
      return null;
    }

    // Create the byte array to hold the data
    byte[] bytes = new byte[(int)length];

    // Read in the bytes
    int offset = 0;
    int numRead = 0;
    while ( (offset < bytes.length)
          &&
          ( (numRead=is.read(bytes, offset, bytes.length-offset)) >= 0) ) {

      offset += numRead;
```

```
        }

        // Ensure all the bytes have been read in
        if (offset < bytes.length) {
            throw new IOException("Could not completely read file " + file.getName());
        }

        is.close();
        return bytes;

    }
```