YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

EBRU SİMÜLATÖRÜ

YÜKSEK LİSANS TEZİ

Hüseyin SAVRAN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

HAZİRAN 2012

YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

EBRU SİMÜLATÖRÜ

YÜKSEK LİSANS TEZİ
Hüseyin SAVRAN
(105105015)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ahmet AKBAŞ

HAZİRAN 2012

YALOVA Üniversitesi Fen Bilimleri Enstitüsü'nün 105105015 numaralı Yüksek Lisans Öğrencisi **Hüseyin SAVRAN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"EBRU SİMULATÖRÜ "** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Prof. Dr. Ahmet AKBAŞ** .....
Yalova Üniversitesi

Jüri Üyeleri : **Prof. Dr. Ahmet AKBAŞ** .....
Yalova Üniversitesi

Doç. Dr. Müfit ÇETİN .....
Yalova Üniversitesi

Doç. Dr. Mustafa ÖZTAŞ .....
Yalova Üniversitesi

Teslim Tarihi : **18 Mayıs 2012**
Savunma Tarihi : **29 Haziran 2012**

Sanatçı ve Mühendislere,

ÖNSÖZ

Tez danışmanım sayın Prof. Dr. Ahmet Akbaş hocama, tez çalışma sürecimde yaptığı değerli katkılardan dolayı sayın Doç. Dr. Müfit Çetin hocama ve çalışma arkadaşlarıma çok teşekkür ediyorum. Ayrıca eşime ve oğlum Hasan Emir'e bu süreçte gösterdikleri anlayış ve hoşgörülerinden dolayı teşekkür ediyorum. Hususen, beni yetiştiren ve zor zamanlarımda yanımda olan babam ve anneme sonsuz minnetlerimi sunarım.

Haziran 2012

Hüseyin Savran

(Bilgisayar Mühendisi)

İÇİNDEKİLER

KISALTMALAR

FPS **:** Saniye Başına Ekran Yenileme Sayısı(Frame Per Second)
GPU **:** Grafik İşlem Ünitesi (Graphical Processing Unit)
SVG **:** Ölçeklenebilir Vektör Grafikleri (Scalable Vector Graphics)
XML **:** Genişletilebilir İşaretleme Dili (Extensible Markup Language)

SEMBOLLER

$\vec{u}$	: Hız vektörü
$\vec{x}$	: Konum vektörü
x	: Konum vektörünün x bileşeni
y	: Konum vektörünün y bileşeni
z	: Konum vektörünün z bileşeni
u	: Hız vektörünün x bileşeni
v	: Hız vektörünün y bileşeni
w	: Hız vektörünün z bileşeni
t	: Zaman değişkeni
p	: Basınç
ρ	: Yoğunluk
∇	: Gradyan operatörü
∂	: Kısmı türev operatörü
$\vec{g}$	: Yerçekimi ivmesi
ν	: Kinematik viskosite
$\nabla.\nabla$	: Laplace operatörü
∇^2	: Laplace operatörü
$\vec{F}$	: Kuvvet vektörü
$\vec{a}$	: İvme vektörü
m	: Kütle
V	: Hacim
d/dt	: Zamana göre türev
D/Dt	: Materyal türev
μ	: Dinamik viskosite katsayısı
q	: Zamana ve konuma bağlı herhangi bir değişken
Dt	: İki frame arasında geçen süre
$\vec{C}$	: 2 boyutlu uzayda yayılan damlanın merkez noktasının konumu
$\vec{P}$	: 2 boyutlu uzayda bir noktanın konumu
$\vec{P'}$	: 2 boyutlu uzayda taşınan noktanın yeni konumu
r	: Yayılmakta olan damlanın ulaşabileceği en büyük yarıçapı
s	: Damlanın Toplam yayılma süresi
k	: Difüzyon oranı
d_orijinal	: Orijinal verilerin tutulduğu iki boyutlu dizi
d_yedek	: Yedek verilerin tutulduğu iki boyutlu dizi
d_o	: Orijinal verilerin tutulduğu iki boyutlu dizi
d_y	: Yedek verilerin tutulduğu iki boyutlu dizi
A	: Komşu düğümlerden sol üst köşedeki düğüm
B	: Komşu düğümlerden sağ üst köşedeki düğüm
C	: Komşu düğümlerden sol alt köşedeki düğüm
D	: Komşu düğümlerden sağ alt köşedeki düğüm
t	: Hız vektörünün boyutlandırma katsayısı

M	**:** Difüzyon sırasında komşu düğümlerden çıkarılacak miktar
Ma	**:** Difüzyon sırasında a düğümünden çıkarılacak miktar
Mb	**:** Difüzyon sırasında b düğümünden çıkarılacak miktar
Mc	**:** Difüzyon sırasında c düğümünden çıkarılacak miktar
Md	**:** Difüzyon sırasında d düğümünden çıkarılacak miktar
a	**:** Geri taşıma için komşu düğümlerden sol üst köşedeki düğüm
b	**:** Geri taşıma için komşu düğümlerden sağ üst köşedeki düğüm
c	**:** Geri taşıma için komşu düğümlerden sol alt köşedeki düğüm
d	**:** Geri taşıma için komşu düğümlerden sağ alt köşedeki düğüm
i	**:** Sütun indisi
j	**:** Satır indisi
fx	**:** Bir noktanın sol üst köşedeki en yakın düğüme yatay uzaklığı
fy	**:** Bir noktanın sol üst köşedeki en yakın düğüme düşey uzaklığı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

EBRU SİMÜLATÖRÜ

ÖZET

Bilgisayarların grafik uygulamalarında daha esnek bir şekilde kullanılmasına olanak sağlayan yazılım ve donanım araçları, günümüzde hızlı bir gelişim sürecine girmiştir. Bu araçlar, karmaşık animasyon süreçlerinin sanal ortamda gerçekleştirilmesi açısından önemli kolaylıklar sağlamaktadır. Bilgisayar teknolojilerinin sağladığı bu kolaylıklardan yararlanarak, geleneksel el sanatlarından 'ebru'nun eğitim ve öğretimi için sanal bir atölye ortamı hazırlanabilir. Bu yolla, ebru sanatının icra edilmesi ile ilgili zorluklar büyük ölçüde aşılabilir.

Literatürde bu amaçla yapılan çalışmalar vardır. Bu çalışmalarda ebru sanatının temel bileşenleri olan tekne, akışkan, boya, biz gibi elemanlar arasındaki etkileşimleri temsil eden matematiksel modellerin basitleştirilmiş şekillerinden yararlanılmaktadır. Başarılı bir ebru simülasyonu için, öncelikle dikkate alınacak ebru çeşidine uygun yapısal özellikleri belirlemek ve bunları kullanılacak modellere yansıtmak gerekir. Oysa, ebru sanatını icra eden ustaların bugüne kadar kullandığı farklı yaklaşımların literatürdeki modelleme çalışmalarına tümüyle yansıdığı söylenemez. Bu kapsamda dikkate alınması gereken ebru çeşitlerinden birisi de, çok köklü bir geleneğe sahip olan ve 'Kadim Ebru' olarak da bilinen 'Battal Ebru'dur.

Bu tez çalışmasında sanal ortamda Battal Ebru desenlerinin üretimi için bir simülatör tasarımı gerçekleştirilmiştir. Bu amaçla kullanıcının ebru araçları ile etkileşimini olabildiğince gerçeğe uygun şekilde yapabilmesine imkân sağlayan bir arayüz tasarlanmıştır. Tekneye tek tek ya da gruplar halinde serpiştirilen boya damlalarının birbirini iterek yayılması süreci, akışkanlar dinamiğini temsil eden Navier-Stokes denklemlerinin basitleştirilmiş şekilleri kullanılarak gerçeklenmiştir. Bu şekilde oluşan ebru çıktılarında her damla için oluşan nihai desen, ölçeklenebilir vektörel grafik (SVG) formatına uygun özel bir kenar yumaşatma işleminden sonra yeniden üretilmiştir. Böylece oluşan ebru çıktılarının, gerçeğine oldukça uygun desenler olduğu gözlenmiş ve bununla ilgili performans değerlendirmeleri yapılmıştır.

MARBLING SIMULATOR

SUMMARY

Nowadays, there is a rapid progress on software and hardware tools to make more flexible graphical productions. These tools provide comprehensive means for achieving complex animation procedures in virtual environment. Taking advantage of such facilities of computer technologies, an artificial environment can be generated to teach and practice marbling, one of the traditional crafts. By this way, limitations while performing marbling craft can substantially suppressed.

There are studies for this purpose in literature. In these studies, simplified forms of mathematical models representing marbling craft's basic components such as tank, marbling liquid, die, and biz are used. For a successful marbling simulation, first of all, structural features that is suitable for related marbling type are required to be determined and such models need to be reflected into models which will be employed. However, it cannot be said that different approaches used by artists performing marbling craft according to studies in the literature until today are entirely modeled in digital environment. In this scope, one of the marbling types which needs to be taken into consideration is Battal Marbling which has a very deep rooted tradition and is known as ‘Kadim Marbling’.

In this study, a simulator for producing Battal Marbling patterns in digital environment is successfully designed. To this end, an interface which enables users to interactively employ marbling tools as much as realistically is created. Through this interface, the process in which the die drops that have been sprinkled either one by one or in groups spread by pushing each other is substantiated via simplified forms of Navier-Stokes equations representing fluids dynamics. In the marbling outcomes which are formed by the mentioned way, the final pattern for each drop is re-produced after a special edge smoothing process which is suitable for Scalable Vector Graphics format (SVG). The acquired marbling outcomes are observed as accurate to the real ones and performance assessments about the results have been accomplished.

1. GİRİŞ

Geleneksel el sanatlarından ebru, icra edilmesi kadar seyir zevki açısından da çok ilgi çeken bir sanattır. Bununla beraber, bu sanatın icra edilmesi için bir kısım zorluklara katlanmak gerekir. Öyle ki, bu sanatı icra edecek bir kişi; tekne, ebru akışkanı, biz, fırça, tarak, boya gibi çeşitli araçlara ve en azından evin bir odasını işgal edecek büyüklükte bir atölye ortamına sahip olması gerekir. Ayrıca üretim için ebru yapma sürecinde kullanılacak olan akışkan ve boyaların hazırlanması, bunların fiziksel özelliklerinin gerekli şartları sağlayacak hale getirilmesi ve diğer ön çalışmalar, bu konularda önceden ustalaşmayı gerektirmektedir.

Ebru sanatının icrası için aşılması gereken bu ön şartların yanında, üretim için hazırlanan bir ebru teknesindeki akışkan ve boyaların bir süre sonra yenilenmesi gerekmektedir. Bu da her seferinde ebru teknesi, akışkan ve boyalarla ilgili hazırlıkların tekrarlanması anlamına gelmektedir. Bütün bu zahmet ve zaman kaybını gerektiren uğraşlar ebru sanatının daha yaygın olarak kullanılmasının önünde engel oluşturmaktadır.

Diğer taraftan, içinde bulunduğumuz bilişim çağının getirdiği yenilikler sayesinde insanlar neredeyse tüm günlük işlerini bilgisayar vasıtasıyla yapmaya başlamıştır. Bunun sayısız faydaları görüldükçe daha çok sayıda insan eliyle yapılan çalışma hızla bilgisayarlar aracılığıyla sanal ortamda gerçekleştirilmektedir. Sanat dallarımız da bu rüzgâra çoktan kapılmıştır. İşte bu noktada geleneksel ve çok eski tarihi olan ebru sanatının bilgisayara taşınması vakti gelmiştir. Bu konuda araştırmacıların şu ana kadar yaptığı bazı çalışmalar bulunmakla beraber, yeni çalışmalara da ihtiyaç duyulmaktadır. Öyle ki, halen normal bir bilgisayar kullanıcısının dahi kolayca kullanabileceği bir ebru yazılımının varlığına tanık olduğumuz söylenemez.

1.1 Motivasyon

Bu güne kadar Ebru simülatörü yapmayı amaçlamış diğer araştırmacıların çalışmalarında gördüğümüz önemli bazı eksikleri bildirelim. Birincisi, Ebru çeşitlerinin en eskisi olan "Battal Ebru" veya "Kadim Ebru" diye isimlendirilen ve sadece fırça ile boya serpiştirmeye dayalı ebru çeşidi göz ardı edilmiş. Bu Ebru çeşidinde boya damlaları tekneye fırça yardımıyla rastgele serpiştirilir. Ortalama üst üste üç veya dört renk boya atılır. Üste atılan boyalar alttakileri kenarlara ite ite mermer kesitinde görülendamarlara benzer yuvarlaklar arasında damarlar oluşur. Usta ebrucunun farkı bunu ne kadar güzel yaptığıyla ölçülür.

Lu ve diğ. [10], Battal Ebruyu modellemişler fakat kullanıcı damlaların yayılışını göremiyor. Atılan damla anında diğerlerini itmiş ve şekil oluşmuş oluyor. Aslında Ebru Sanatında mühim bir nokta, Ebru süreci sonunda üretilen resmin güzelliği yanında, sanatçının bunu üretirken aldığı zevk de düşünülmelidir. O yüzden başarılı bir Ebru yazılımında damlaların yayılışını, sanatçı bilgisayar ekranında bizzat görüp zevk etmelidir. Aynı durum karıştırrırken de geçerlidir. Gerçi karıştırma konusunda bir kaç tane başarılı yazılım geliştirilebilmiştir. Fakat bunları ortalama bir bilgisayarda çalıştırıp bir çıktı üretmek mümkün değildir.

Bu tez çalışmamızda Battal Ebrusunu aslına uygun olarak ortalama bir bilgisayarda yapılabilmesini sağlayacak bir yazılımın geliştirilmesi amaçlanmıştır. Bunu yaparken bizden önceki çalışmalarda kullanılan yoğun işlem gerektiren algoritmalar yerine onları akıllıca basitleştirerek ve temelde sıvıların basınç özelliğinden yola çıkarak bir algoritma geliştirilmesi düşünülmektedir. Ayrıca, Lu ve diğ. [10]'nin gerçek zamanlı olmayan algoritmasını zamanın bir fonksiyonu haline getirerek, Battal Ebru simülasyonu için ikinci bir yöntem geliştirilmesi de düşünülmektedir.

Geliştirilecek algoritmaları desteklemek için, bu algoritmaların ürettiği çıktıların makyaj adımı diyebileceğimiz son bir işlemden sonra gayet yumuşak eğrisellik gösterebilen SVG [21] formatına çevrilmesi planlanmıştır.

1.2 Tezin Amacı

Bu çalışmada ortalama bir bilgisayarda çalışabilecek, insanların en azından ebru ile tanışmasını sağlayabilecek bir simülasyon yazılımının tasarlanması amaçlanmıştır. Ebru simülatörü yapmak 2004 yıllarında başlayan çok yeni bir konu olmasına karşın gerekli algoritmaların geliştirilmesinde Euler [1], Navier-Stokes [2] gibi yöntemlerin kullanıldığı görülmektedir. Navier ve Stokes soyadlı iki fizik bilim adamı, 1822 yıllarında akışkanlar fiziği konusunda tabiattaki çoğu akışkanı modelleyebilen Navier-Stokes [2] denklemlerini bulmuşlardır. Fakat bu denklemler sadece bazı basit vakaların analitik çözümlenmesinde kullanılmıştır.

Araştırmalarda bilgisayar teknolojileri kullanılmaya başlanması ile birlikte araştırmacılar denklemleri çözmek için sonlu farklar, sonlu hacimler, sonlu elemanlar ve spektral yöntemleri gibi nümerik algoritmalar geliştirmişlerdir. Genel olarak bu algoritmalar oldukça karmaşık ve işlem süresi açısından uzun zaman alıcı olmasına rağmen, daha iyi ve doğru sonuçlara ulaşmaya çalışmaktadır. Dolayısıyla özellikle gerçek hayatta genellikle çok önemli projelerde kullanılmaktaydılar. Örneğin bir uçağın kanadındaki ya da bir köprüdeki gerilimin tam değerleriyle hesaplanmasının ne kadar kritik olduğu açıktır.

Bilgisayar grafiğinde ve bilgisayar oyunlarında en önemli unsurlardan biri algoritmaların çok karmaşık olmamaları, aksine hızlı ve mümkün olduğu kadar doğal sanal gerçeklik sağlamalarıdır. Böylece bilgisayar oyunları, standart PC, oyun konsolları ve mobil cihazlarda çalışabilirler. Bu amaca ulaşmak için literatürdeki geleneksel fizik hesaplamalarından ziyade, görsel etkiler oluşturmak için Stam [3,4] gibi özel dikim algoritmalar geliştirilmiştir.

Ebru simülasyonunun gerçeklenmesinde de benzer zorluklar bulunmaktadır. Bilindiği üzere ebru teknesi içindeki akışkanın ve üzerine damlatılan boyaların etkileşimi moleküler düzeyde gerçekleşmektedir. Bilgisayarda bire bir moleküler düzeyde modellemek ve gerçek zamanlı çalıştırmak günümüz işlemci kapasiteleri ile mümkün değildir. Ayrıca ebrunun simülasyonu için kullanılan Navier-Stokes [2] denklemlerini çözebilmek için nümerik analiz yapmak gerekir.

Bu denklemlerin lineer olmayan diferansiyel denklem olmaları da başka bir problemdir. Çünkü gerçek zamanlı olarak kısıtlı bir çözünürlük ve boyuttaki teknede algoritmayı çalıştırmak zorunda kalırız. Daha büyük tekne boyutlarına ihtiyaç duyulursa algoritmaların optimize edilmesi problemiyle başa çıkmamız gerekir.

Bir diğer durum ise denklemler ebruyu modellemek için yeterli değildir. Ebru sanatının doğal olarak gerçeklenmesinde bazı önemli fiziksel ve kimyasal şartlar söz konusudur. Mesela tekne içine damlatılan boyanın yayılması, yayılma hızı ve gücünün boya içindeki öd ve su oranına göre değişmesi, tekne içindeki iki damlanın aynı özellikleri de taşısalar kesinlikle birleşip karışmaması ve biz yardımıyla boyanın karıştırılması gibi özellikler buna örnek verilebilir.

Bu çalışmada ebru sanatı hakkında bilgiler çoğunlukla Derman [14], Derman [15], Dinçer [16] referanslarından faydalanılmıştır. Ebrunun tarihçesi, kullanılan malzemeler, yapılışı ve ebru çeşitleri konusunda çok aydınlatıcı olmuşlardır.

Akışkanların dinamiği konusunu iyice anlamak için Bridson [17] ve Butts [18] kaynaklarından faydalanılmıştır. Bu kaynaklarda Navier-Stokes [2] denklemleri analitik olarak incelenmiştir. Bu denklemler ayrıntılı bir şekilde incelendikten sonra ebru için gerekli olan kısımları belirlenip nümerik analiz yapılarak uygulanmıştır.

Yazılım dili olarak C++ tercih edilmiştir. Grafik kütüphanesi olarak DirectX [19] kütüphanesi kullanılmıştır. Ayrıca West [20]'nin akışkanlar dinamiğini uygularken Stam [4]'in yöntemini geliştirerek "Geri Taşıma" ismiyle bir yöntem eklemiştir. Bunları uygulamak için geliştirdiği kaynak kodlar başlangıç şablonu olarak çalışmamızda kullanılmıştır.

1.3 Literatür Araştırması

Hesapsal Akışkanlar Dinamiği konusunda literatürde birçok yayın olmakla birlikte tek başına akışkanlar dinamiği çalışmaları ebru simülasyonu için yetersiz kalmaktadır.

Dolayısıyla özellikle ebru simülatörü geliştirmeye yönelik çalışmalar literatür çalışması olarak incelenmiştir. Literatürdeki çalışmalar dikkate alındığında ebru konusunda 2004 yılından günümüze bir kaç çalışma yapılmıştır.

Öncelikle ebru simülatörü tasarlama yolundaki köşe taşlarından birisi olan Jos Stam [3] bu tez çalışmasında çok faydalı olmuştur. Navier-Stokes [2] denklemlerini ve bunları bilgisayarda çözmek konusunu anlamamız açısından bu çalışmasında akışkan dinamiğine ait çoğu fiziksel olayı açıklamaktadır. Bu fiziksel olayların bilgisayarda çözümüne yönelik temel teşkil edecek algoritmaları [3] numaralı çalışmasında açıklamıştır. Hem anlatmış hem de örnek kodlarını vermiştir. Jos Stam [4]'in "oyunlar için gerçek zamanlı akışkan dinamikleri" konulu yayını, bu tez çalışması ve tüm oyun programcıları için önemli bir çalışmadır. Çünkü Navier-Stokes [2] denklemlerinin bilgisayar oyunları için kullanımının önünü açmıştır.

Çiğdem ve diğ. [5], ebrunun gerçek fiziksel yapısı hakkında bir çalışma yayınlamışlardır. Ebru teknesinin içinde ne olup bitiyor bir fizikçi gözüyle incelemişlerdir. Yaptıkları çalışmada boya, su, öd ve kitreden oluşan malzemelerin yoğunlukları, yüzey gerilim sabitleri ve karışım oranlarını belirlemişlerdir. Ayrıca boya karışımlarının kitreli su yüzeyinde birbiriyle etkileşimlerini inceleyerek hareket hızları ve yüzeyde oluşturdukları tabaka kalınlıklarını hesaplamışlardır. Bu tabaka kalınlıklarının azami değerlerinin aşımında oluşan boya çökmelerini incelemişlerdir. Bu araştırmadan, tez çalışmasında öd ve boya maddelerinin çeşitli oranlarda karıştıklarında gösterdikleri hareket hızlarının nicelik ve nitelikleri konusunda faydalanılmıştır.

Acar ve diğ. [6], Ebru konusunda doktora çalışması yapmışlardır. Bu çalışma, dijital ebru için jenerik bir araç olarak mezoskala dinamiklerine ve akışkan sıvı denklemlerine dayanan çok boyutlu bir sıvı modeli sunar. Ayrıca Acar [7], doktora tez çalışmasında sayısal bir ebru için çok önemli bilgiler vermektedir. Tez çalışmamızda önemli bir şablon ve ebrunun sayısal modeli hakkında bilgi kaynağı olmuştur. Bu geleneksel ebru tekniklerinin yanı sıra, laminer ve türbülanslı akımları simüle etmek için çeşitli kullanıcı kontrolü sağlar. Malzeme taşıma, son derece ayrıntılı, keskin sıvı ara yüzlerine sahip ebru desenlerini ele alabilmek için geliştirilmiş ileri bir adveksiyon çözümüne dayanmaktadır.

Jin ve diğ. [8], ise Navier-Stokes denklemlerini grafik işlem ünitesi(GPU) üzerinde çözdürüp performans açısından çok kazanım sağladılar. Renklendirme kısmında ise her rengi bir katmana koyarak farklı renkleri ayrı ayrı işleme sokmuşlardır. Bir ebru çalışması yapılırken en fazla 8 çeşit renk kullanılacağını tespit etmişler. Örneğin, birinci katmanda açık mavi renkli bir serpiştirme yapıldığında, farklı bir katman seçip orada da sarı renkli damlalar serpilebilmektedir. Son görüntüyü oluşturulurken, katmanlar bir araya toplanıp tek katmana dönüştürülür.

Ando ve diğ. [9], ise kontur tabanlı bir algoritma geliştirmişlerdir."Vektör Akışkanı" diye isimlendirdikleri yöntemde sıvı içinde katı bir bölge kapalı bir kontur olarak temsil edilir ve ebruya benzer bir kıvırcık ve net şeklini koruyarak sıvı akışı tarafından taşınır. Konturu meydana getiren noktalar ebru içinde gördüğümüz renk öbeklerinin çevresinin çizilmesinde kullanılır. Kıvrım sayısı ne kadar artarsa nokta sayısı da artar ve sistemin işlemsel yükünü artırır. Net kıvrımlar çizmek ve direk vektörel çıktı vermek açısından iyi bir yöntemdir. Fakat belirli bir karmaşıklıktan sonra performans açısından düzgün çalışmamaktadır.

Lu ve diğ. [10], Navier-Stokes [2] denklemlerinden tamamen uzaklaşarak "Matematiksel Ebru" diye isimlendirdikleri yeni bir metot geliştirmişlerdir. Bu metotla birlikte diferansiyel denklemlerle hiç uğraşmadan basit matematiksel fonksiyonlarla ebruya ait fiziksel özellikler modellenebilmektedir. Örneğin, damlaların birbiri içinde veya yan yana genişleyerek desen oluşturduğu Battal Ebru, daireyle ilgili basit eşitliklerle rahatlıkla gerçeklenebilir. Yani bir damla, ulaşması gereken çapa ulaşınca etrafındaki damlaların piksellerini kendi yarıçapıyla orantılı olarak taşır. Hem teknedeki karıştırma işlemleri, sinüs dalgalarına benzediği için basit bir sinüs fonksiyonu ile gerçekleştirilebilir. Buna göre sinüs dalgasının geçtiği koordinatlara uzaklığıyla ters orantılı olarak her bir piksel dalga yönünde taşınarak yeni şekil oluşur. Bu yeni algoritma gayet basit, hızlı ve etkili olmasına karşın önemli bir eksikliği vardır. Bu eksiklik, algoritmanın gerçek zamanlı olmamasıdır. Yani damlayı tekneye serptiğinizde damla aniden büyüyüp, diğerlerini itmiş oluyor. Damlanın yayılma sürecini sanatçı göremiyor. Karıştırma süreci de aynı şekilde aniden olup bitiyor. Tez çalışmamızda bu algoritmanın Battal Ebru'yla ilgili kısmı zamanın bir fonksiyonu olarak yeniden düzenlenip uygulanmıştır.

Ebru teknesinde yayılan damlaların düzgün eğrilere benzemesi, birçok yazılım paketinde eğri çizdirme yöntemi olarak kullanılan Bezier eğrilerinin [11] ebru desenlerinin çizdirilmesinde de işe yarayabileceğini akla getirmektedir. Kübik Bezier [11] eğrisi, Bezier eğri çeşitlerinden birisidir. Bu eğri çeşidinde her bir eğri noktasının iki adet kontrol noktası vardır. Bu kontrol noktalarının ait oldukları eğri noktalarına göreceli pozisyonlarına göre iki nokta arasındaki eğrisellik özelliği belirlenir. Klasik olarak bu kontrol noktaları polinom fonksiyonlarla belirlenir. Dolayısıyla biraz karmaşıktır. Inkscape [12] yazılım paketi Kübik Bezier [11] eğrisi çiziminde kullanılan açık kaynak kodlu bir yazılımdır. Bu yazılım paketinde eğrilerin daha sürekli ve yumuşak bir şekilde çizilebilmesi için kontrol noktalarını otomatik düzenleme araçları vardır. Bu araçlarda kullanılan yöntem, Spiro Eğrileri [13] denilen bir yöntemdir. Bu yöntem tez çalışmamızda, Inkscape [12] yazılımının kaynak kodlarından örnek alınarak, aynen kullanılmıştır. Çünkü algoritması gayet basittir.

Ebru konusunda yapılmış başka bir çalışma da Xu ve diğ. [24]'e aittir. Bu çalışmalarında Ebruda kullanılan çiçek figürleri gibi figürleri gayet iyi üretebilen GPU kullanan bir yazılım geliştirmişlerdir. Daha çok Jin ve diğ. [8]'in çalışmasını bir adım ileriye taşımışlardır.

Zhao ve diğ. [25], büyük boyutlardaki teknelerde bile ebru yapmaya imkân veren ve GPU kullanan bir yazılım geliştirmişlerdir. Bu çalışmalarında özellikle Navier-Stokes [2] denklemleri GPU sayesinde hızlı çözerek avantaj sağlamışlardır. Ayrıca sayısal dağılmayı azaltmak için üçüncü dereceden bölünmemiş yarı-Lagrage kısıtlı İnterpolasyon Profili yöntemi uygulamışlardır.

Ando ve diğ.[9], bu çalışmalarını genişlettikleri başka bir çalışmalarında [26], yeni bir yüzey izleme algoritması geliştirerek vektörel çıktı üretmişlerdir.

Tüm tanıttığımız çalışmalarda gördüğümüz ortak ve çok önemli eksik, Kadim Ebru veya Battal Ebru diye bilinen Ebru'nun ilk ve en güzel formlarından olan sadece boya damlatılarak yapılan ebru çeşidini modellememiş olmalarıdır. Yani hiç bizle karıştırmadan önce düşük ödlü boyalar serpiştirip, üzerine biraz daha yoğun ödlü, biraz daha derken bir mermer kesitini andıran görünüm oluşuyor. Bizce bu, gerçekten önemlidir. Çünkü karıştırma kısmında sanat değerinin farkı tam anlaşılmazken Battal ebruda renklerin uyumu öd miktarının dengesi gibi faktörler ürünün sanat değerini ortaya koyuyor. Bu yüzden Battal Ebrusu belki ilk önce modellenmesi gereken olgudur. Daha sonra biz aletinin etkisi modele eklenebilir.

1.4 Tezin Kapsamı

Tezin ilk bölümünde çalışmada yakalamak istediğimiz hedef ve literatür özeti verilmiştir. İkinci bölümde ebru hakkında biraz bilgi paylaşılmıştır. Bu bölümde ebrunun tarihçesi kısaca anlatılmıştır. Ebruda kullanılan malzemeler tanıtılmıştır. Bu malzemelerle yapılabilecek ebru çeşitlerine resimler eşliğinde kısaca değinilmiştir. Üçüncü bölümde, Navier-Stokes [2] denklemlerini ve bunlara dayalı akışkan altyapısı anlatılmaya çalışılmıştır. Dördüncü bölümde ise Navier-Stokes sistemini ebru'ya has fizikleri dahil ederek kendi yöntemimizle yeniden düzenlenmiştir. Geliştirilen yazılım da bu bölümde tanıtılmıştır. Son bölümde de özet ve çalışmanın kritiği verilmiştir.

2. GELENEKSEL EBRU

Bu bölümdeki bilgilerin derlenmesinde yoğun bir şekilde Derman [14], Derman [15], Dinçer [16] referanslarından faydalanılmıştır.

2.1 Ebru'nun Tarihçesi

Ebru Sanatının ne zaman nerede ortaya çıktığı net bir şekilde bilinmemesine karşın bugün dünyanın neredeyse her yerinde bilinmektedir. Tarih boyunca da her toplum ebruya kendine has özellikler kazandırmıştır. Ebru Genel olarak doğu milletlerine özgü bir süsleme sanatı olduğu düşünülmüştür. İrandaki bazı eski kaynaklarda Ebrunun İran'a Hindistandan geçtiği yazılıdır. Bazılarında ise Türkistandaki Buhara kentinde doğup buradan irana oradan da Osmanlıya geçtiğinden bahsedilir. "Ebru" kelimesi "buutumsu" veya "kaş" manasına gelir. Gerçekten de en eski ebru çeşidi olan Battal Ebrusu bulutsu ve kaşımsı desenler içerir. Avrupa ise Ebruyu Osmanlıdan öğrenmiştir. Ebru batıda "Mermer Kâğıt" veya "Türk Kağıdı" olarak bilinmektedir. Hatta ingilizcede "Marbling" yani mermerleme anlamında kullanılmaktadır. Bunda ebru deseninin mermere de benzemesinin mutlaka payı vardır.

Kitaplarda kaplama malzemesi olarak kullanılagelmesinden anlaşılan kağıdın gelişmesiyle paralel bir gelişim çizgisine sahiptir Ebru. Bir tarihsel rakam vermek gerekirse ebrunun başlangıcı 9. ve 10. yüzyıllara kadar götürülebilir. Eski alim ve yazarlar el emeği göz nuru, dimağların parıltısı kitaplarını ebru süsüyle taçlandırmışlardır. Bugün hala kitap kapağı veya iç taraftaki ilk ve son sayfalarda süsleme amaçlı kullanılır. Hattatlar yine yazılarının arka planı olarak en güzel ebru desenlerini kullanırlar [15].

Osmanlı döneminde başlı başına bir sanat ve iş kolu olan ebruculuk, 20. yüzyıl başlarına gelindiğinde unutulma noktasına gelmiştir. Bu sanatın tekrar hayat kazanması, ebru sanatına 'çiçekli ebru'yu geliştiren büyük sanatçı Necmeddin Okyay sayesinde olmuştur. Okyay'dan sonra yeğeni Mustafa Düzgünman, Türk Ebrusunda teknik zirve ve kaliteyi yakalamış ve bir çok usta ebrucu yetiştirmiştir. Hocası Okyayın çiçekli ebrularına papatyayı eklemiştir. Lale, menekşe, gül, papatya desenli ebrular, Kadim Ebruyu ikinci plana atacak kadar itibar kazanmıştır.

2.2 Ebru Malzemeleri ve Hazırlanışı

Ebru için gerekli malzeme aşağıdaki gibi sıralanabilir [15]:

50 cm x 35 cm boyutunda çinko, çelik , galveniz veya tahtadan bir tekne, sıvıyı tutar. 50'ye 35 oranı korunmak kaydıyla daha küçük veya büyük boyutlar da seçilebilir. Mesela 25 cm x 17 cm, ve 100 cm x 70 . sıvı derinliği 6 cm dolaylarında olabilir.

Bu 50 cm x 35 cm boyutundaki bir teknedeki boyayı emebilecek A3 boyutunda sarı veya beyaz kâğıtlar. Tekne içindeki sıvı, suyla kitrenin uygun oranda karıştırılmasıyla oluşur. Kitre çoğunlukla geven bitkisinin gövdesi çizilerek oradan akan ve katılaşan krem renkli sert maddeden ya da bir tür yosun olan deniz kadayıfından elde edilir. Geven bitkisi kullanılacaksa suda 5 veya 6 gün bekletilmesi ve arada bir karıştırılması gerekir. Deniz kadayıfı is bir günde istenilen neticeyi verir. Acelesi olan daha kısa bir sürede de idare edecek bir kıvamı tutturabilir.

Sığır ödü, boya ve sudan oluşan homojen Ebru boyası ebrunun en temel malzemesidir. Boyanın rengi tabiatta bulunan özel renkli toprakların veya kayaların iyice ezilmesiyle elde edilir. Bazı boyalar da bitkilerden elde edilir. Bu boyaların temel özelliği suda erimemesi ve yağ içermemesidir. Sığır ödü, sığırın safra kesesindeki sıvının kaynatılmasıyla elde edilir. Gayet kötü bir kokusu vardır. Fakat Ebruyu ebru yapan öddür denebilir. Ebru boyasının sıvı yüzeyde çökmeden dağılabilmesini ve diğer damlalarla karışmamasını sağlayan öddür. Asıl ustalık boyalardaki öd oranını yaralayabilmeye bakar. Tekneye atılan ilk damlalar etrafları boş olduğundan rahat dağılırlar. Fakat bunların üzerien atılan damlalar alttaki boyayı etrafa iterek kendine yer açar. Üste atılan damladaki öd oranı daha fazla olmalı ki çökmesin ve dağılabilsin.

Boyayı tekneye ilave etmek için iki yöntem vardır. Birincisi fırça ikincisi “biz “ aletidir. Fırçayı ebrucunun sarması bir gelenektir. Hatta tüm ebru malzemelerini ebrucunun hazırlaması beklenir. Fakat artık bunlar hazır satın alınabilmekte ve çoğu kişinin de kendisi hazırlaması pek mümkün olamamaktadır. Fırça, yaşlı atların kuyruklaının takriben 10 cmlik demetler halinde 20-30 cm uzunluğundaki gül dallarına olta ipiyle sarılıp bağlanmasıyla elde edilir. Boyanın tekneye çoklu damlalar halinde rastgele serpiştirlmesi fırça ile yapılır. Biz aleti ise tornavidaya benzer sapı tahta, ucu ise çeşitli kalınlıklarda ince silindir şeklinde metalden oluşur.

Boya kavanozuna batırılan biz, hemen arkasından tekneye dik olarak batırılır ve boya istenen alana ulaşıncaya kadar beklenir. Ve çıkarılır. Biz ile teknedeki boya karıştırılarak desen ve çiçek figürleri yapılır. Bizlerin türevi sayabileceğimiz taraklar ise bir çok biz ucunun bir tahataya istenen aralıklarda montajıyla elde edilir. Biz ile Battal ebru dışındaki tüm ebru çeşitlerinde biz ve taraklar kullanılır. Tüm ebru çeşitleri yapılmadan önce fırça ile zemin rengi ve deseni mutlaka verilir. Bir de bazı serpmeli ve battal ebrularında serpilen boyalara üç boyutlu küreye benzer görünüm kazandıran neft maddesi kullanılır.

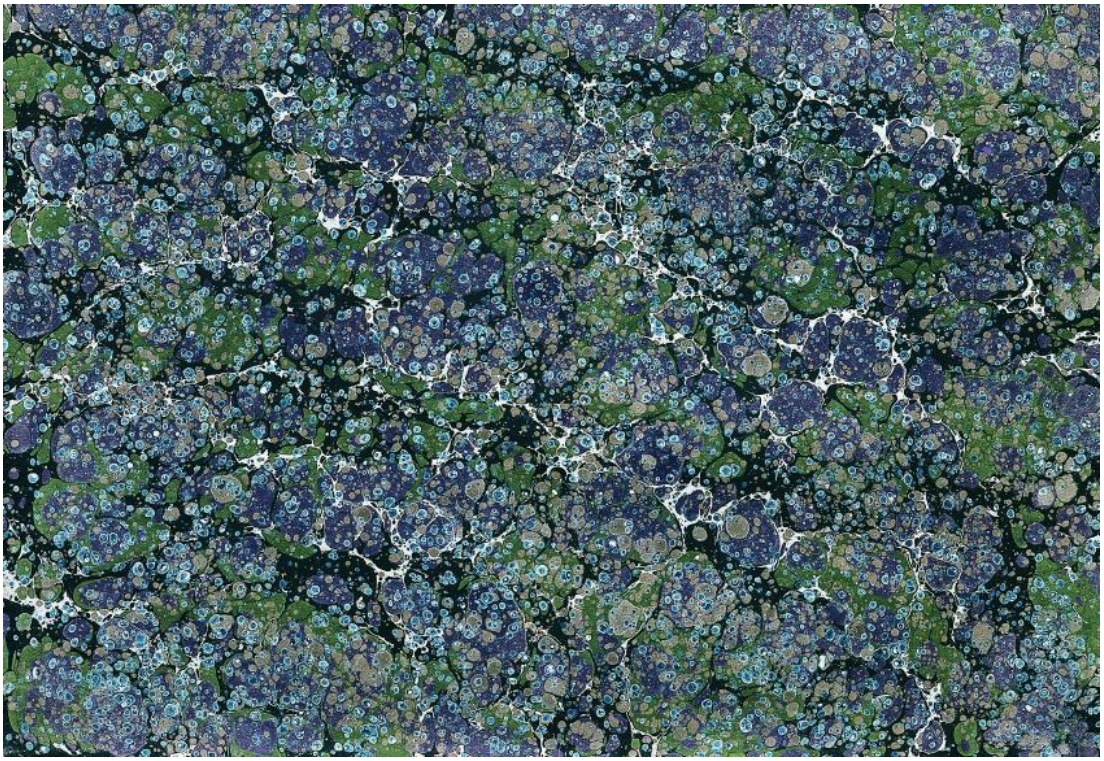
Ebrunun meşhur renkleri vardır. Bu renklerle ebru yapıldığında daha hoş olmaktadır. Fakat ebru sanatçısı elde edebildiği her türlü renkle ebru yapabilir. Tabiatta bulunan arsenik sülfürden sarı renk, lahor çividinden mavi, sarı ve mavi karışımından yeşilin tonları, bedahşi laciverti denen doğal çivit, isten elde edilen siyah, İsfidaç bazik karbonatın tabiattaki şeklinden beyaz, demir oksitli topraktan kırmızı, çamlıca toprağından tütün rengi gibi ve bunların tonları yaygın olarak kullanılmaktadır. Son yıllarda her renk sentetik pigment boya da özellikle çiçikli ebrularda kullanılmaktadır.

Tek tek her ebru çeşidinin nasıl yapıldığı konusu bu tezin kapsamı dışında olduğu için şimdilik kısa kesiyoruz. Bu konuda ebru çeşitleri başlığında kısaca bilgi verilecektir.

2.3 Ebru Çeşitleri

2.3.1 Battal Ebru

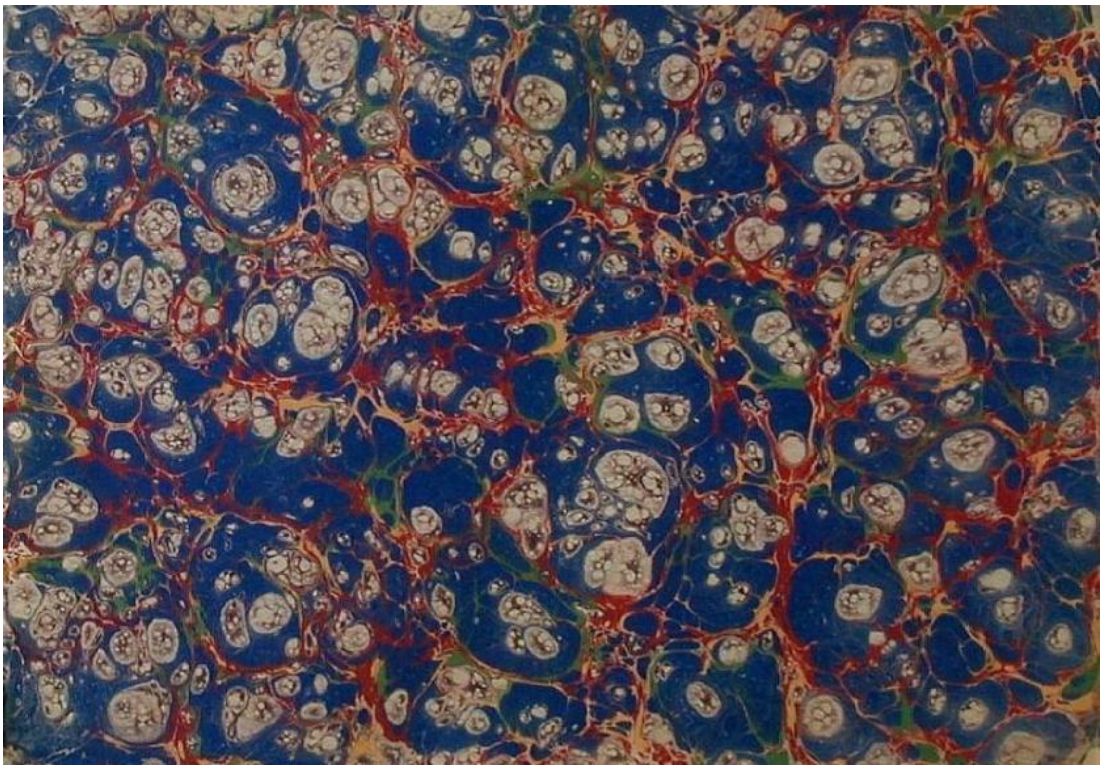
Boyaları fırça ile tekneye atarak ve tarak, biz gibi malzemeler kullanılmadan yapılan ebrudur. Bilinen en eski ebru çeşididir ve bu yöntemle yapılan battal ebrulara Tarz-ı Kadim (eski tarz) battal ebru denir. Çünkü zamanla daha değişik tarzlarla da battal ebrular yapılmıştır. Boyaları tekne üzerine fırça ile atmak kolay iş gibi görülebilir ancak görüldüğü kadar kolay değildir ve ustalık gerektirir. Ebrucunun ustalığı battal ebrularından belli olur. Merhum Mustafa Düzgünman'ın ifadesiyle "Battal Ebru ebrunun ilk mektebidir". Somaki battal, neftli battal, serpmeli battal gibi değişik çeşitleri vardır.



Şekil 2.1 : Mustafa Düzgünman'ın bir battal ebru çalışması

2.3.2 Gelgit Ebru

Atılan battal ebruya bir biz daldırılır ve bütün tekne, birbirine paralel hareketlerle aşağı yukarı veya sola sağa gezilir. Ortaya çıkan gelgit ebruya çizgileri dik olarak kesecek şekilde tekrar aynı gel git işlemi uygulanırsa buna da taramalı ebru denir. Taramalı ebru da bir gel git ebru çeşididir.



Şekil 2.2 : Mustafa Düzgünman'ın bir neftli battal ebru çalışması

Şekil 2.3 : Alparslan Babaoğlu'na ait serpmeli gelgit ebru

2.3.3 Şal Ebru

Gelgit ya da taramalı ebru yapıldıktan sonra bizle, daha düzensiz ve dairemsi hareketler yapılarak şal desenine benzeyen şal ebru elde edilir. Biz hareketleri makul miktarda yapılmalıdır aksi takdirde boyalar çamurlu bir görünüm almaya başlar.

2.3.4 Serpme

Yukarıda anlatılan tüm ebru çeşitlerinin üzerine serpme yapılabilir. Bu durumda ebrular serpmeli battal, serpmeli gel-git, serpmeli şal gibi isimler alır. Serpme işlemi için neftli boyalar veya çamlıca toprağı gibi açık renkli boyalar tercih edilir. Fırça elle çok iyi bir şekilde kavanoza sıkılır ve boya yüksekten sert darbelerle serpilir.

2.3.5 Bülbül Yuvası

Git gide küçülen damlalar şeklinde atılan battal ebru üzerinde yapılır. Bizle dıştan içe doğru helezonlar çizilir. Genellikle teknenin uzun kenarı boyunca 5-6, kısa kenarı boyunca 4-5 helezon yapılır. Gel-git ebru veya taraklı ebru üzerinde de bülbül yuvası çalışılabilir.

2.3.6 Taraklı Ebru

Gelgit ebru veya taraklı ebru yapıldıktan sonra, son yapılan gelgite dik olarak tarağın ucu tekneye daldırılır ve sabit bir hızla teknenin bir ucundan diğer ucuna doğru çekilir. Bu işlem tarak çıkartılmadan bir de tersi yöne yapılırsa buna ters taraklı ebru denir. Taraklı ebru yapıldıktan sonra üzerine çok ince bir bizle gelgit ebru, şal ebru veya serbest hareketler yapılabilir.

2.3.7 Zemin Ebrusu

Çiçekli veya hatip ebruya zemin oluşturan ebru çeşididir. Aynı boyanın az ödlüsü, çok ödlüsü ve neftlisi hazırlanır. En alta az ödlü, onun üzerine çok ödlü, en üste de serpme tekniğiyle neftli boya atılır. Zemin ebrusu üzerinde çalışılacak motiflerin öne çıkması için genelde açık tonlarda çalışılır.

2.3.8 Hafif Ebru

Battal, taramalı ve taraklı ebrunun kıvamı daha sulu olan sıvı üzerinde, normalden daha sulu ve ödlü boyalarla çalışılmasıyla yapılır. Hafif ebrulu kâğıtlar üzerine hat ve tezhip çalışmaları yapılır ve yazı yazılır.

2.3.9 Dalgalı Ebru

Hemen her çeşit ebru ile dalgalı ebru yapmak mümkündür. Kâğıdın bir kısmı teknenin ucuna yerleştirilir ve kâğıt ileri geri hareket ettirilerek teknede dalga oluşması sağlanır, bir yandan kâğıdın bu hareketine devam edilir bir yandan da yavaş yavaş kâğıt tekneye kapatılarak oluşan dalgalar yakalanır. Dalgalı ebruya sarhoş ebru veya tercih etmesek de İspanyol ebru isimleri de verilmiştir.

Şekil 2.4 : Mustafa Düzgünman'a ait bir bülbül yuvası

2.3.10 Kumlu Ebru ve Kılçıklı Ebru

Teknedeki sıvı kullana kullana kirlenerek öyle bir kıvama gelir ki atılan boyalar istense de istenmese de kum gibi nokta nokta bir görüntü almaya başlar. Sadece kumlu ebru yapımına battal ebru ile başlanmaz. Bir damlalıkla tekneye yakın mesafeden aynı nokta veya noktalara boya damlatılır. Boya yayılmaya başlar. Kumlu ebru için Sacid Okyay'ın buluşuyla sığır ödü yerine kalkan balığı ödü kullanılması daha iyi sonuç vermiştir. Ancak bu maddenin kokusu sığır ödünden çok daha kötüdür. Kumlu ebru yapılırken noktalar daha da irileşip V şeklini almaya başlarsa buna da kılçıklı ebru denir. Boya olarak genellikle lahor çividi tercih edilir. Boya az sulu ve ödlü olarak hazırlanır.

2.3.11 Çift Ebru

Daha önce yapılmış bir ebrulu kâğıdın üzerine yeni yapılmış bir ebru alınarak yapılır.

2.3.12 Hatip Ebru

18 yy. da Ayasofya Camii Hatibi Mehmed Efendi tarafından bulunduğu için bu ismi almıştır. Zemin üzerine çiçekli veya hatip ebru için hazırlanmış boyalardan belli aralıklarla birer damla bırakılır. Daha sonra her damlanın oluşturduğu dairenin içine ikinci, üçünce ve isteğe göre daha fazla sayıda damla bırakılır ve iç içe değişik renklerden oluşturulmuş daireler elde edilir. Genellikle teknenin uzun kenarı boyunca 5-6, kısa kenarı boyunca 4-5 daire oluşturulması tercih edilir. Daha sonra bir bizle bu dairelerin içinde soldan sağa, yukarıdan aşağıya, çapraz hareketler yapılarak hatip desenleri elde edilir. Yürekli, taraklı yürek, yıldız, çarkıfelek, menekşe bu desenlere verilen çeşitli isimlerdir. Hatip desenlerinin hepsinin ya da bazılarını aynı ebru üzerinde çalışmaya Hatip-i Mütenevvia denir.

2.3.13 Çiçekli Ebru(Necmeddin Ebrusu)

Hatip ebrunun icadından sonra ebruda çiçek yapılmasına da çalışılmış ancak fazla başarı sağlanamamıştır. 1918 yılından itibaren merhum Necmeddin Okyay çiçek çalışmalarını ıslah ederek lale, karanfil, hercai menekşe, gelincik, gonca gül, kasımpatı, sümbül gibi çiçekleri doğal şekline en yakın şekilde resmetmeyi başarmıştır. Onun yetiştirdiği merhum Mustafa Düzgünman da bu tarza papatyalı ebruyu ilave etmiştir. Çiçek yapımında önce damlatılan yeşil boyalardan sap ve gövdeler, daha sonra da bu sap ve gövdelerin uygun noktalarına bırakılan diğer renklerden çiçekler yapılır. Çiçek ve hatip yapımında kullanılan boyaların çok iyi terbiye edilmiş olması ve diğer ebru çeşitlerinde kullanılan boyalara göre daha koyu bir kıvama sahip olmaları gerekir. Ancak böylelikle boya damlaları şekil vermek için bizle lastik gibi uzayarak çekilebilir ve arzu edilen yerde bırakılarak istenen motifler elde edilebilir.

Çiçekli ebrular Necmeddin Okyay'ın talebesi Ord. Prof. Dr. Süheyl Ünver'in teklifi üzerine sanat tarihimizde Necmeddin Ebrusu ismiyle anılmaktadır. Merhum Okyay'ın Üsküdar yeni Camii imamı olması sebebiyle hatip ebrusuna karşılık çiçekli ebruların da imam ebrusu olarak anılması Reisülhattatin Hacı Kamil Akdik (1861-1941) tarafından teklif edilmişse de Necmeddin Ebrusu denilmesi daha uygun bulunmuştur.

2.3.14 Akkase Ebru

Eski kitap sanatları içinde, bir kağıdın yazı yazılacak kısmının ayrı, etrafının ayrı renge boyanmasına akkase, böyle kâğıtlara da akkaseli kâğıt denir. Bu sanat ebruya da tatbik edilmiştir. Kağıdın yazı yazılacak kısmının kendi renginde kalması istenirse, oraya Arap zamkı sürülür. Kâğıt kuruduktan sonra tekneye yayılırsa zamklı kısmın dışında kalan yerle ebrulanmış olur, Arap zamkı bulunan bölge ise kendi renginde kalır boyayı kabul etmez. Buna akkaseli ebru denir.

Ebruda başka bir tarz akkase şöyle yapılmaktadır. Daha önce tarifini yaptığımız hafif ebrulu kağıdın, ortada yazı yazılacak kısmına Arap zamkı sürülerek, kuvvetli (koyu) renklerle hazırlanan bir ebru teknesine tekrar yayılmasıyla zamklı kısımlar bu ikinci ebruyu tutmaz. Arap zamkı sürülmüş kısım kendi hafif ebrulu haliyle, diğer kısımlar ise üst üste gelen çift ebrulu haliyle kalır. Böylece iki ayrı ebrulu, yani ebrulu akkase denen kâğıt yapılmış olur. Hafif ebrulu kağıda zamkla yazı yazıp, koyu renkli tekneye tekrardan yaymak suretiyle, merhum Necmeddin Hoca akkaseyi yazılı ebruya da tatbik etmiştir. Ebruda akkase uygulaması eskiden kâğıt yapıştırma usulü ile yapılırken daha sonra yazılı ebruda olduğu gibi arap zamkı kullanmak yoluna gidilmiştir.

3. AKIŞKANLARIN MODELLENMESİ

Literatürde akışkan mekaniği ve akışkan dinamiğini temsil eden analitik modeller, akışkanların sahip olduğu viskozite ve yoğunluk gibi parametreler ile basınç ve hız gibi değişkenler arasındaki ilişkinin zamana ve uzaysal parametrelere bağlı olarak nasıl değiştiğini belirleyen Navier-Stokes kısmi diferansiyel denklemleriyle temsil edilmektedir [2]. Bu tez çalışmasına konu olan akışkanlar, esasen ebru teknesi içinde suya 'kitre' maddesinin ilave edilmesi suretiyle oluşturulan 'taban akışkanı' ve 'ebru boyası'dır. Her iki akışkanın sahip olduğu ortak fiziksel özellik, bunların "sıkıştırılamaz" nitelikte olmasıdır. Aşağıda bu tür akışkanların dinamiğini temsil eden Navier-Stokes denklemlerinin analitik ifadeleri irdelenerek bu tez çalışmasında kullanılacak sayısal model için bir taban oluşturulmaya çalışılmıştır.

3.1 Analitik Akışkan Modelleri

Burada sıkıştılamaz niteliğe sahip akışkanların dinamiğini temsil eden analitik modeller incelenirken, ilgi sahamızı oluşturan ebru akışkanlarına ait hareketliliğin olabildiğince düşük hızlarda gerçekleştiği hususu dikkate alınmıştır. İncelemelerde akışkan değişkenleri ve parametreleri gerçek analitik incelemeler için olması gereken moleküler (mikroskopik) boyut yerine makroskopik boyutta dikkate alınmıştır.

Buna göre akışkanın davranışını karakterize eden temel fiziksel değişkenler akışkan hacminin olabildiğince küçük bir parçasını temsil eden hacim birimleri (akışkan parçacığı) bazında tanımlanmıştır. Öyle ki, her akışkan parçacığı 3-boyutlu uzayda x, y, z bağımsız değişkenleri ile temsil edilen bir konuma sahiptir. Akışkan parçacığına eşlik eden değişken ve/veya parametrelerin zamanla değişimi söz konusu olduğunda dördüncü boyutu belirleyen bağımsız değişken olarak t zaman değişkeni kullanılmıştır. Teorik irdelemeler 3-boyutlu model üzerinde yapılmış olmakla beraber, ebru uygulaması için akışkanın üçüncü boyutu ihmal edilerek x, y konum değişkenleri ile temsil edilen 2-boyutlu uzay (düzlem) esas alınmıştır.

Bu tanımlamalara göre akışkanın dinamiğini temsil eden temel fiziksel değişkenlerden birisi $\vec{u}(x,y,z,t)$ sembolü ile temsil edilen hız değişkenidir. Bu değişken, akışkanın işgal ettiği hacim içerisindeki her noktada 3 kartezyen skaler bileşen (u,v,w) ile temsil edilen bir vektörel değişkendir. Benzer şekilde basınç, $p(x,y,z,t)$, akışkanın herhangi bir konumda sahip olduğu basıncı temsil eden 1 boyutlu skaler bir değişken; yoğunluk, $\rho(x,y,z,t)$, akışkanın herhangi bir konumda sahip olduğu yoğunluğu temsil eden 1 boyutlu skaler bir parametredir.

Böyle bir akışkanın dinamiği esas olarak 'kütlenin korunumu', 'momentum dengesi' ve 'enerjinin korunumu' ile ilgili temel fiziksel kanunlara bağlı olarak belirlenir. Öyle ki, bu kanunların akışkan dinamiği ile ilgili yorumları aşağıdaki gibi sunulabilir [18]:

- Kütlenin korunumu: Akışkanın toplam kütlesi, akışkan parçacıkları bazındaki kütle birimlerinin hız alanları boyunca yayılması ile değişmez,
- Momentum dengesi: Akışkana uygulanan iç ve dış kuvvetlerin toplamı her an maddenin momentumundaki değişime eşittir,
- Enerjinin korunumu: Akışkanın iç enerjisiyle kinetik enerjisinin toplamı daima sabittir.

Buna göre Navier-Stokes denklemleri, sıkıştırılamaz nitelikteki akışkanlar için en genel haliyle aşağıdaki kısmı diferansiyel denklemlerle verilir [18]:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u}.\nabla)\vec{u} + \frac{1}{\rho}\nabla p = \vec{g} + \nu\nabla.\nabla\vec{u} \qquad \textbf{(3.1)}$$

$$\nabla.\vec{u} = 0 \qquad \textbf{(3.2)}$$

Burada $\vec{u}$ akışkanın hız vektörünü, p akışkan basıncını, ρ akışkan yoğunluğunu, $\vec{g}$ yerçekimi ivmesini, ν ise kinematik viskositeyi (akışkanlık) temsil etmektedir. ∇, gradyan operatörü olarak bilinen $(\frac{\partial}{\partial x}\ \frac{\partial}{\partial y}\ \frac{\partial}{\partial z})$ vektörünü, $\nabla.\nabla$ ise ∇^2 şeklinde de sembolize edilebilen Laplace operatörünü temsil etmektedir.

Bunlardan yoğunluk, akışkanın örneğin su olması halinde yaklaşık 1000 kg/m^3 iken, hava olması halinde 1.3 kg/m^3 civarındadır. Yerçekimi ivmesi aşağı doğru ve 9.81 m/s^2 değerindedir. Viskosite ya da akışkanlık, akışkanın karıştırılma zorluğunu temsil eden bir parametredir. Bu parametre, akışkanın akarken biçim değiştirmeye ne kadar dirençli olduğunun bir ölçüsünü temsil etmektedir. Örneğin bal ve pekmez yüksek akışkanlığa sahip iken su ve alkol düşük akışkanlığa sahiptir. Ebru boyasının akışkanlığı, boya içine katkılanan 'öd sıvısı' ile ayarlanır.

3.2 Momentum Denklemleri

Yukarıda (3.1) eşitliği ile verilen denklem akışkanın momentum dengesinin uzaysal konuma ve zamana bağlı olarak nasıl değiştiğini göstermektedir. Bu denklem Newton'un ikinci yasası olan $\vec{F} = m\vec{a}$ eşitliğindeki *m* kütleli bir cisim üzerine uygulanan kuvveti belirleyen $\vec{a}$ ivme vektörünün akışkan şartlarında nasıl belirlendiğini göstermektedir. Burada dikkate alınan akışkanın sıkıştırılmazlık şartı (3.2) eşitliğiyle temsil edilmektedir.

Buna göre, (3.1) eşitliği, $\vec{F} = m\vec{a}$ eşitliğinin daha karmaşık bir yorumu ile dikkate sunulabilir. Akışkanın toplam hacminin olabildiğince küçük akışkan parçacıklarından oluştuğu ve her parçacığın bir *m* kütlesi, bir *V* hacmi ve bir $\vec{u}$ hız vektörü ile temsil edildiği varsayılırsa, her parçacık üzerine gelen kuvvet vektörlerinin değişim şekli (3.1) eşitliğiyle verilen analitik ifadeye göre belirlenir. Dolayısıyla (3.1) eşitliği, her bir parçacığın, üzerine etkiyen kuvvetlerle nasıl hareketleneceğini ya da nasıl ivmeleneceğini belirler.

$\vec{F} = m\vec{a}$ eşitliğinde kullanılan ivme değişkeni, $\vec{a}$, gerçekte Navier-Stokes akışkan modellerinde *d/dt* türev operatörü yerine, "materyal türev" olarak adlandırılan ve *D/Dt* şeklinde temsil edilen operatör kullanılarak aşağıdaki gibi temsil edilmektedir:

$$\vec{a} = \frac{D\vec{u}}{Dt} \qquad \textbf{(3.3)}$$

Burada *D/Dt* materyal türev operatörü aşağıdaki gibi tanımlanmaktadır [18]:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u}.\nabla = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z} \qquad \textbf{(3.4)}$$

Bu türev, ilgili değişkenin zamana ve uzaysal konuma bağlı olarak hız vektörlerince nasıl değiştirildiğinin belirlenmesi için kullanılır. Buna göre, Newton'un ikinci yasası Navier-Stokes denklemlerinde aşağıdaki gibi temsil edilir.

$$m\frac{D\vec{u}}{Dt} = \vec{F} \tag{3.5}$$

Her bir akışkan parçacığı üzerine etkiyen kuvvet vektörlerini belirlemek için (3.1) eşitliğinin her iki tarafındaki terimleri, akışkan parçacığının m kütlesi ile çarpıp, eşitliğin sol tarafındaki ilk iki terim yerine (3.5) eşitliği ile verilen kuvvet ifadesini yerleştirdiğimizde, (3.1) eşitliği aşağıdaki biçime dönüşür [18]:

$$m\frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla p + V\mu\nabla.\nabla\vec{u} \tag{3.6}$$

Bu eşitliğin sağ tarafındaki terimlerden ilki olan m$\vec{g}$ akışkan parçacığı üzerine etkiyen yerçekimi kuvvetidir. Diğer kuvvetler ise, ilgili konumdaki akışkan parçacığı üzerine komşu akışkan parçacıkları tarafından uygulanan kuvvetleri temsil eder.

Buna göre, (3.6) eşitliğinin sağ tarafındaki ikinci terim basınç kaynaklı kuvveti temsil etmektedir. Basınç kuvveti akışkan parçacıklarının birbiri üzerine yaptığı basıncın dengesizliğiyle oluşur. Bir akışkan parçacığının sahip olduğu basınç komşu akışkan parçacığının sahip olduğu basınçtan büyükse, bu yüksek basınçlı parçacıktan düşük basınçlı parçacığa doğru yönelmiş bir kuvvetin oluşumuna neden olur. Bir akışkan parçacığı üzerinde oluşan basınç dengesizliği basıncın negatif türevi (-∇p) alınarak bulunur. Bunun akışkan parçacığının temsil ettiği hacim boyunca entegre edilmiş hali, yaklaşık olarak akışkan parçacığının V hacmi ile çarpılmak suretiyle bulunur.

(3.6) eşitliğinin sağ tarafındaki üçüncü terim ise akışkanlıktan kaynaklanan kuvveti temsil eder. Öyle ki, her bir akışkan parçacığı çevresindeki akışkan parçacıklarını ortalama hızla hareket etmeye zorlar. Yani her akışkan parçacığının çevresindeki akışkan parçacıklarıyla olan hız farkını en aza indirmeye çalışır. Buna göre, bir değişkenin ortalamdan ne kadar uzaklaştığının ölçüsü Laplace operatörü $\nabla.\nabla = \nabla^2$ ile bulunur. Bu akışkan parçacığının hacmi boyunca entegre edilip dinamik akışkan sabiti μ ile çarpıldığında akışkan kuvveti elde edilir.

Diğer taraftan, (3.6) eşitliği ile belirlenen değişken değerlerinin akışkan parçacıklarının boyutuna bağlı olarak bir hata payı içerdiği açıktır. Bu durumda hata payını azaltmak için akışkan hacmini oluşturan parçacıkların saysısını sonsuza götürerek limit değer hesaplandığında seçilen akışkan parçacıklarının hacmi sıfıra gider. Bu durumda *m* kütlesi ve *V* hacmi de sıfıra gideceğinden bir sorun ortaya çıkar.

Bu sorun, eşitliğin her iki tarafındaki terimleri *V*'ye bölüp limitini alarak giderilebilir. *m*/*V*'nin her durumda akışkanın yoğunluğunu (ρ) temsil ettiği bilindiğinden, (3.6) eşitliği aşağıdaki gibi yeniden düzenlenebilir:

$$\rho \frac{D\vec{u}}{Dt} = \rho\vec{g} - \nabla p + \mu\nabla.\nabla\vec{u} \qquad \textbf{(3.7)}$$

(7) eşitliğinde bu kez eşitliğin her iki tarafındaki terimler yoğunluğa bölünerek aşağıdaki eşitlik elde edilir:

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + \frac{\mu}{\rho}\nabla.\nabla\vec{u} \qquad \textbf{(3.8)}$$

(8) eşitliği biraz daha basitleştirilerek kinematik vizkosite $\nu = \mu/\rho$ olarak alınırsa eşitlik son olarak aşağıdaki şekle dönüşür:

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + \nu\nabla.\nabla\vec{u} \qquad \textbf{(3.9)}$$

Böylece (1) eşitliği ile verilen Navier-Stokes denklemleri, bilgisayar grafiklerinde akışkanın modellenmesi açısından daha kullanışlı olan *D*/*Dt* materyal türev ile ifade edilen forma kavuşmuş olur.

3.3 Lagrange ve Euler Yaklaşımları

Bilgisayarlı grafik uygulamalarında akışkanlar modellenirken, çeşitli fiziksel değişkenlerin hız vektörleri tarafından taşınması ile ilgili yorumlar yapabilmek açısından *D*/*Dt* materyal türev ifadesinin yer aldığı Navier-Stokes denklemlerini kullanmak daha elverişlidir. Materyal türevin bu avantajını daha iyi anlamak için, Navier-Stokes denklemlerinin çözümü açısından farklı iki yaklaşımı sergileyen Lagrange ve Euler bakış açılarını yakından tanımakta fayda vardır.

Lagrange yaklaşımı, sürekli bir akışkan ortamını parçacıklara ayırarak inceler. Buna göre akışkan içindeki her bir (x, y, z) konumu bir $\vec{u}$ hız vektörüne sahip ayrı bir parçacık olarak etiketlenir ve izlenir. Öyle ki, gerektiğinde her parçacık boyutu yeterince küçültülerek, akışkanın bir molekülüymüş gibi değerlendirilebilir.

Euler yaklaşımı ise, her bir parçacığı izlemek yerine uzaysal konumları sabitlenmiş noktalarda yoğunluk, sıcaklık ve hız gibi ölçülen büyüklüklerin zamanla nasıl değiştiğine odaklanır.

Euler yaklaşımında akışkanın dinamiğini yansıtan değişken veya parametreler, akışkanın geçiş yaptığı varsayılan bu sabit uzaysal konumlar için belirlenir. Öyle ki, her geçiş ilgilenilen değişkenin o sabit noktadaki değişimine bir katkı sağlar. Örneğin bir sabit noktadan sıcak akışkan geçtikçe o noktadaki sıcaklık artarken, geçip giden akışkanın sıcaklığı sabit kalabilir. Aynı sabit noktadan soğuk akışkan geçmeye başladıkça akışkanın genel sıcaklığı artıyor olsa bile o noktadaki sıcaklık azalabilir.

Her iki yaklaşım balonla yapılan bir hava durumu izleme uygulaması örneği ile açıklanabilir. Şöyle ki, anılan izleme Lagrange yaklaşımına göre, balon ile havada rüzgar boyunca hareket ederek balonun bulunduğu konumdaki havanın basınç, sıcaklık ve nem gibi değerlerini ölçerek yapılır. Euler yaklaşımında ise, sabit bir yerde duruyorken etraftan geçen havanın basınç, sıcaklık ve nem değerleri ölçülür. Nümerik olarak, Lagrange yaklaşımı parçacık sistemine karşılık gelirken Euler yaklaşımı zamanla konumu değişmeyen sabit ızgara yapısına karşılık gelir..

Euler yaklaşımının karmaşık olmasına rağmen tercih ediliyor olmasının iki nedeni vardır. Birincisi, basınç gradyanı ve viskosite gibi uzaysal türev gerektiren ifadeler üzerinde analitik irdelemeler yapmak bu yaklaşımla kolaylaşmaktadır. İkincisi o uzaysal türevleri Euler ızgarasında nümerik olarak belirlemek daha kolaydır.

Navier-Stokes denklemlerini oluştururken Euler ve Lagrange yaklaşımları birlikte değerlendirilir. Bu birliktelikte anahtar kavram materyal türevdir. Buna göre, Lagrange yaklaşımı ile başlanacak olursa, akışkan her biri bir $\vec{x} = (x, y, z)$ uzaysal konumuna ve $\vec{u}$ hız vektörüne sahip olan parçacıklardan oluşmaktadır.

Şimdi her parçacığın yoğunluk, sıcaklık, renk veya başka herhangi bir değişkeni temsil eden genelleştirilmiş bir q değişkenine eşlik ettiğini varsayılsın. Buna göre, $q(t,\vec{x})$ fonksiyonu q değişkeninin $\vec{x}$ konumunda t anında geçerli olan değerini belirler. Dolayısıyla burada q değişkeni bir Euler değişkenidir. Lagrange yaklaşımı ise, q değişkeninin o parçacık için değişim hızının ne olduğu ile ilgilenir. Bunun cevabı için zincir kuralı kullanılarak türev alınacak olursa, materyal türeve ulaşılır [18]:

$$\frac{d}{dt}q(t,\vec{x}) = \frac{\partial q}{\partial t} + \nabla q.\frac{d\vec{x}}{dt} \tag{3.10}$$

$$\frac{d}{dt}q(t,\vec{x}) = \frac{\partial q}{\partial t} + \nabla q.\vec{u} \tag{3.11}$$

$$\frac{d}{dt}q(t,\vec{x}) \equiv \frac{Dq}{Dt} \tag{3.12}$$

Görüldüğü gibi (3.11) eşitliğinin sağ tarafındaki ifade materyal türevin ifadesidir. Buna göre, (3.11) eşitliğinin sağ tarafındaki $\partial q/\partial t$ terimi sabit konumu verilen bir uzaysal noktadaki q değişkeninin değişim hızını temsil etmektedir (Euler ölçümü). İkinci terim $\nabla q.\vec{u}$ ise, bu değişimin geçmişte bu noktadan geçmiş bulunan hız vektörleri tarafından katkılanan kısmını temsil etmektedir (Lagrange yaklaşımı). Bu durum, bir termometredeki sıcaklığın, termometreye çarpan hava parçacıklarındaki sıcaklığın değişmesinden dolayı değil, oraya çarpıp geçen farklı sıcaklıktaki parçacıklardan dolayı gerçekleştiğine ilişkin bir örnekle açıklanabilir.

Toplam ifadeyi daha açık görmek için (3.4) eşitliği ile verilen materyal türev ifadesi q değişkeni için aşağıdaki gibi yazılabilir:

$$\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} + v\frac{\partial q}{\partial y} + w\frac{\partial q}{\partial z} \tag{3.13}$$

(3.13) eşitliği, q değişkenine eşlik eden akışkan parçacıklarının $\vec{u}$ hız alanı tarafından taşındığına işaret etmektedir. Bu olay literatürde ‘adveksiyon’, ‘konveksiyon’ veya ‘transport’ kavramları ile ifade edilebilen fiziksel gerçekliktir. Bu olay Lagrange bakış açısından q değişkeninin parçacıklara eşlik ettiği ve onunla beraber konum değiştirdiği ve fakat kendisinin değişmediği anlamına gelir. Bu durum, akışkanın dış etkenlerle uyarılmadığı durumdaki denge haline karşılık gelir ve materyal türevin sıfıra eşitlenmesi ile temsil edilir [18]:

$$\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + \vec{u}.\nabla q = 0 \tag{3.14}$$

3.4 Ebru'nun Analitik Modeli

Ebrunun analitik modeli literatürde (3.14) eşitliği ile verilen materyal türev teriminin yanı sıra, basınç, viskosite, yer çekimi ve dış uyaranların da etkisini dikkate alarak ebru akışkanının hareketine eşlik eden hız, yoğunluk gibi çeşitli değişkenlerin nasıl bir değişim gösterdiğini izah eden Navier-Stokes denklemeleri ile verilmektedir [17]. Buna göre biz veya fırça gibi bir dış uyaran vasıtasıyla katkılanan $\vec{\boldsymbol{f}}$ ivme vektörü ile viskosite teriminin de dikkate alınmasıyla hız vektörlerinin nasıl değiştiğini izah eden analitik model aşağıdaki gibi verilebilir [4,18]:

$$\frac{D\vec{u}}{Dt} = \nu\,\nabla^2\vec{u} + \vec{f} \qquad \textbf{(3.15)}$$

Benzer şekilde ebru teknesine biz veya fırça vasıtasıyla serpiştirilen boya akışkanının enjeksiyon hızı ($\boldsymbol{s}$) ile viskosite teriminin de dikkate alınmasıyla, damlatılan boya akışkanının zaman ve konuma bağlı olarak nasıl yayıldığını izah eden analitik model de aşağıdaki gibi verilebilir [4,18]:

$$\frac{D\rho}{Dt} = \kappa\,\nabla^2\rho + s \qquad \textbf{(3.16)}$$

(3.15) ifadesine göre, ebru akışkanının belli bir andaki durumu hız vektörleri alanı şeklinde modellenebilir. Bu modelden ebrunun ayrık (sayısal) modeline geçildiğinde, model uzaydaki her noktaya bir hız vektörü iliştirir ve çok küçük zaman adımlarında hızın nasıl değişeceğini belirler. Bir hız vektörleri alanı, beraberinde ebru boya maddesini hareket ettirinceye kadar tek başına görsel olarak ilgi çekici değildir. Ebru boya maddesinin hareketi, etrafındaki hız alanlarının oluşturduğu kuvvetler tarafından sağlanır. Boya maddesi hız vektörlerini takip ederek yayılır.

(3.16) eşitliği benzer değişimleri yoğunluk parametresi için gündeme taşır. Buna göre yoğunluğun her zaman adımındaki değişimine etki eden üç farklı olay vardır. Bunlardan ilki materyal türev ifadesinde verildiği gibi hız alanıdır. İkincisi (3.16) eşitliğinin sağ tarafındaki ilk terimle ifade edilen difüzyon olayıdır. Üçüncüsü ise, biz veya fırça ile damlatılan ebru boyasının oluşturduğu dış kaynaklı yoğunluktur.

(3.14), (3.15) ve (3.16) eşitlikleri birlikte düşünüldüğünde Stam'in [4] de makalesinde kullandığı gibi daha düzenli bir biçimdeki (3.17) ve (3.18) eşitlikleri elde edilir. (3.17) eşitliği hızın zamanla nasıl değiştiğini ifade ederken, (3.18) eşitliği kütlenin zamanla nasıl değiştiğini ifade eder [18].

$$\frac{\partial u}{\partial t} = -\vec{u}.\nabla\vec{u} + \nu\,\nabla^2\vec{u} + \vec{f} \qquad \textbf{(3.17)}$$

$$\frac{\partial \rho}{\partial t} = -\vec{u}.\nabla\rho + \kappa\,\nabla^2\rho + s \qquad \textbf{(3.18)}$$

Buna göre, bu tez çalışmasında amaçlanan Battal Ebru desenlerinin oluşumuna eşlik eden sayısal model aşağıda açıklandığı gibi oluşturulmuştur.

4. EBRU'NUN SAYISAL MODELİ

Navier-Stokes [2] denklemleri gerçek zamanlı olmayan ve gerçek değerlere yakın değerlerin lazım olduğu hesaplamalarda çok kullanılmakla beraber gerçek zamanlı çalışan sistemlerde nadiren kullanılmaktadır. Çünkü hesap edilecek sıvı boyutu büyüdükçe denklemin nümerik çözüm boyutu üssel olarak büyümektedir. Ebru gibi gerçek zamanlı olması gereken akışkan modellerinde aynen kullanılması işe yaramamaktadır. Fakat yine de yoğunluk adımındaki tüm denklemleri biraz basitleştirerek ve hız denklemleri yerine başka bir yöntem kullanarak gerçek zamanlı bir model kurabiliriz. Bu bölümde modelimizin kuruluşu anlatılmaktadır.

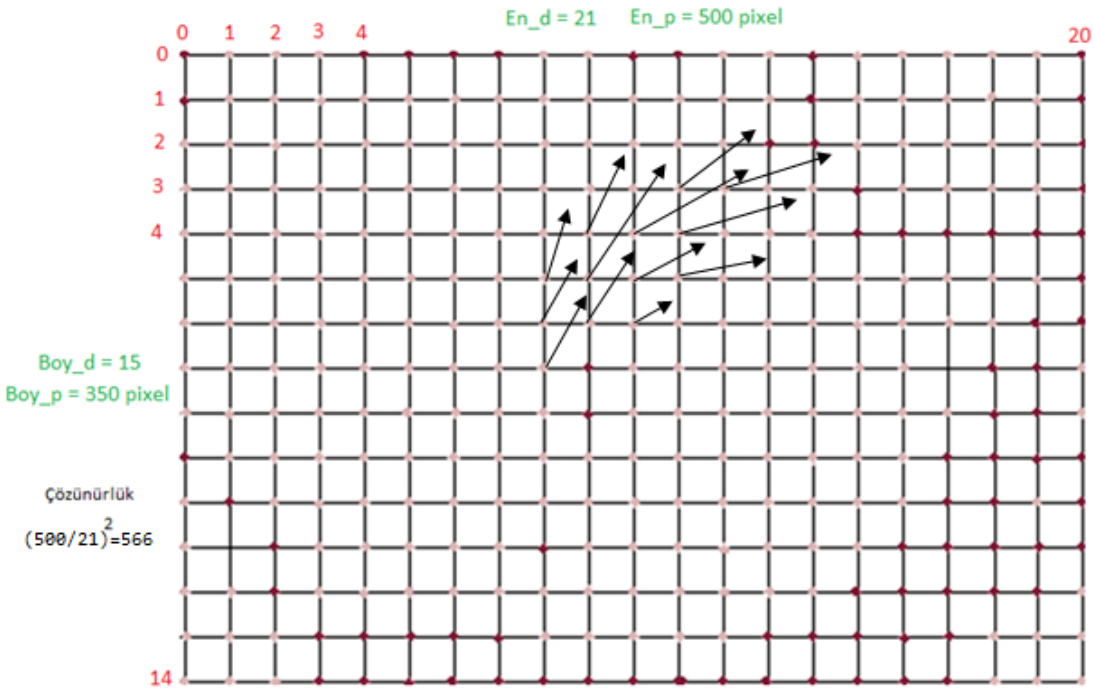
4.1 Ebru Teknesinin Sayısal Analizi

Bu kısımda ebru teknesini, boyaları ve öd kavramlarını bilgisayarda nasıl modellediğimizi göstereceğiz.

4.1.1 Teknenin Sayısal Modeli

Ebru teknesi içindeki yaklaşık 6 cm derinliğindeki sıvının bir hacmi olmasına karşın, boyalar bu hacmin çok küçük bir kısmını kullanır. Boyalar suyun üzerinde kalıp kendi hacimleri kadar bir yer kaplarlar. Hacim deyince üç boyutlu bir ortam akla geliyor. Mesela tekneye damlatılmış tam daire şeklindeki bir boya damlasını bir merkezi, bir yarıçapı ve bir yüksekliği olan bir silindir gibi hayal edebiliriz. Bu silindirin yüksekliği ihmal edilebilecek kadar küçüktür. Bu yüzden hesaplarda gereksiz yere masraf yapmaması için tekneyi üç boyut yerine iki boyutlu bir düzlem ve bu düzlemde yayılan iki boyutlu damlalar şeklinde düşünebiliriz.

Boyalardaki her molekülü birebir temsil ederek işlem yapabilmemiz performans açısından mümkün olmadığına göre, bunu iki boyutlu bir ızgara şeklinde düşünebiliriz. Izgaranın kesişim yerlerinde de madde ve hız vektörlerinin bilgisini tutabiliriz [4,6]. Şekil 4.1'de sayısal tekne modelimiz gösterilmiştir.



Şekil 4.1 : Ebru teknesinin sayısal modelinin gösterimi.

Kesişim yerlerinin her birini "düğüm" olarak adlandırırız. Satırdaki düğüm sayısına En_d, sütundaki düğüm sayasına Boy_d diyelim. Toplam düğüm sayısı bu iki sayının çarpımıyla bulunur. Bilgisayar ekranında son çıktıyı gösterirken piksel tabanlı çalışmak zorundayız. Bu sefer ikinci bir en ve boy karşımıza çıkıyor. Bu da teknenin ekranda görünen en ve boyudur. Bunlara En_p ve Boy_p diyebiliriz. Normal bir ebru teknesinin en ve boy oranı olan 50 cm x 35 cm'dir. Sayısal teknenin en ve boy oranını normal bir ebru teknesinin oranına göre ayarlamak gerekir. Kullanılacak bilgisayarın gücüne göre düğüm veya tekne boyutlarında oynama yaparak istenilen genişlik ve çözünürlükte sayısal tekne elde edilebilir. Çözünürlük ise En_p'nin En_d'ye bölümünün karesi alınarak bulunabilir. Çözünürlük rakamı toplam piksel sayısını toplam düğüm sayısına bölündüğünde de bulunabilir.

Şekil 4.1'deki örnek sayısal teknenin başlangıç noktası sol üst köşedir. İşlem, sol üst köşeden başlayarak her bir düğüm üzerinden satır sonuna kadar gider, sonra alt satıra geçerek devam eder. Deneysel olarak belirlediğimiz ve geliştirdiğimiz yazılımda ön tanımlı olarak gelen tekne boyut değerleri Çizelge 4.1'de görülebilir.

Çizelge 4.1 : Yazılımda kullanılan tekne boyutları.

En_d	Boy_d	En_p	Boy_p	Çözünürlük
239 düğüm	167 düğüm	743 piksel	520 piksel	9.66 (piksel/ düğüm)

Çizelge 4.1'de çözünürlük sütunu diyor ki her bir düğüm 9 piksel kadar alanı temsil ediyor. Şekil 4.3'deki çözünürlük değeri ise çok daha düşük. Burada her bir düğüm yaklaşık 529 pikseli temsil eder. Yani 23 piksel x 23 piksel'lik bir karenin tam ortasında bir düğüm bulunmaktadır. Çizelge 4.1'de ise 3 piksel x 3 piksel'lik bir karenin ortasına bir düğüm denk gelmektedir. Her bir düğümde tutulan bilgiler şöyledir: madde miktarı, hız vektörü, düğümün ait olduğu damla. Madde miktarı skalar bir niceliktir. Hız vektörü ise orijini ait olduğu düğüm olmak üzere x ve y bileşenlerinden oluşur. Madde miktarı negatif değer alamazken, hız vektörünün x ve y bileşenleri negatif, sıfır veya pozitif değer alabilir. İki düğüm arasındaki mesafe 1 birim varsayılır. Düğüm pozisyonları ve hız vektörleri buna göre belirlenir.

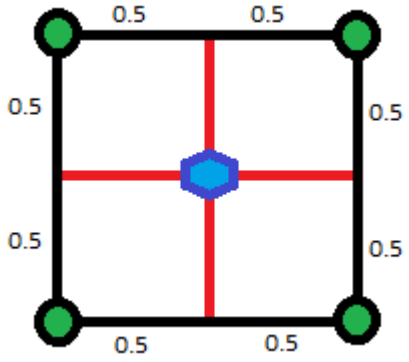
Tekneye atılan her bir damla için yeni bir damla nesnesi oluşturulur. Her bir düğüm mutlaka ve sadece bir damlaya aittir. Damlalarda temel olarak kütle miktarı, öd miktarı ve renk değerleri tutulur. Öd miktarı her bir damlanın özkütlesini bildirir. Özkütleden yola çıkarak damlanın kaplayabileceği azami alan, kütlenin özkütleye bölümüyle bulunur. Damlanın herhangi bir anda kaplamakta olduğu düğüm saysına hazır alan denilir. Kütlenin hazır alana bölümü güç(basınç) değerini verir. Güç, damlanın diğer damlalarla etkileşime girdiğinde devreye girer. Aynı damlaya ait düğümler arasındaki etkileşimde ise öd değeri kullanılır. Bunların nasıl kullanıldığı sonraki bölümlerde ayrıntılı anlatılacaktır.

4.1.2 Zaman Ölçütü ve Kullanıcı Etkileşimi

Görüntü çizdirme ve kullanıcı etkileşimini Microsoft firmasının Directx [19] isimli üç boyutlu grafik kütüphanesini kullanılarak yapılmaktadır. Programlama dili olarak C++ dili kullanılmaktadır. Görüntü çizdirme ve kullanıcı girdisi alma noktasında Directx'e sıkı sıkı bağlı değiliz. Geliştirilen algortima rahatlıkla OpenGL [22] kütüphanesine de uyarlanabilir.

Gerçek zamanlı üç boyutlu simulasyon motorlarında, bunlara oyun motoru da denir, “Frame Per Second (FPS)” yani saniyede ekran tazelenme hızı denilen bir kavram vardır. Mesela FPS değeri 30 olduğu farz edilse, saniyede ekrandaki görüntü 30 kez tazeleniyor demektir. FPS ne kadar büyük olursa algoritma o kadar hızlı demektir. Ekrana çizilen görüntüye “frame” denir. Directx [19], her ekran görüntüsü çizdirme işleminden sonra, ona kaydedilen bir fonksiyonu veya metodu çağırır. Böylece her ekran çiziminden önce mevcut yapıyı güncellemek için kontrol bize geçiyor demektir. Tek tek her düğümden soldan sağa ve yukarıdan aşağı doğru ilerleyerek hız vektörlerine göre madde taşınımları ve gerekli diğer işlemler yapılır. Bu ne kadar hızlı yapılırsa FPS o kadar büyük olacak demektir.

Güncellemeleri zamanın bir fonksiyonu haline getirebilmek için her çizdirme sonrasında kontrol bize geçtiğinde geçen süreyi tespit etmek gerekir. Bu süreyi FPS değerinin çarpmaya göre tersini yani 1’in FPS değerine bölümüyle elde edebiliriz. FPS değeri de Directx [19] kütüphanesinden otomatik elde edilebilmektedir. İki “frame” arasında geçen süreye Dt diyebiliriz. Dt değerini kendimiz de hesap edebiliriz. Mesela kontrol bizden çıkarken zamanı kaydedip tekrar kontrol bize verilince, şimdiki zamanı önceki kaydedilen zamandan çıkarırsak Dt değerini buluruz. Bunun çarpamaya göre tersi ise FPS değerini verecektir. Çalışmamızda Dt süresi böyle hesaplanmıştır.



Şekil 4.2 : Fare sol tuşuna basıldığında fare işaretçisi hangi dört düğümün arasına denk geliyorsa bunların tam ortası hesaplanıp boya maddesi buraya verilir.

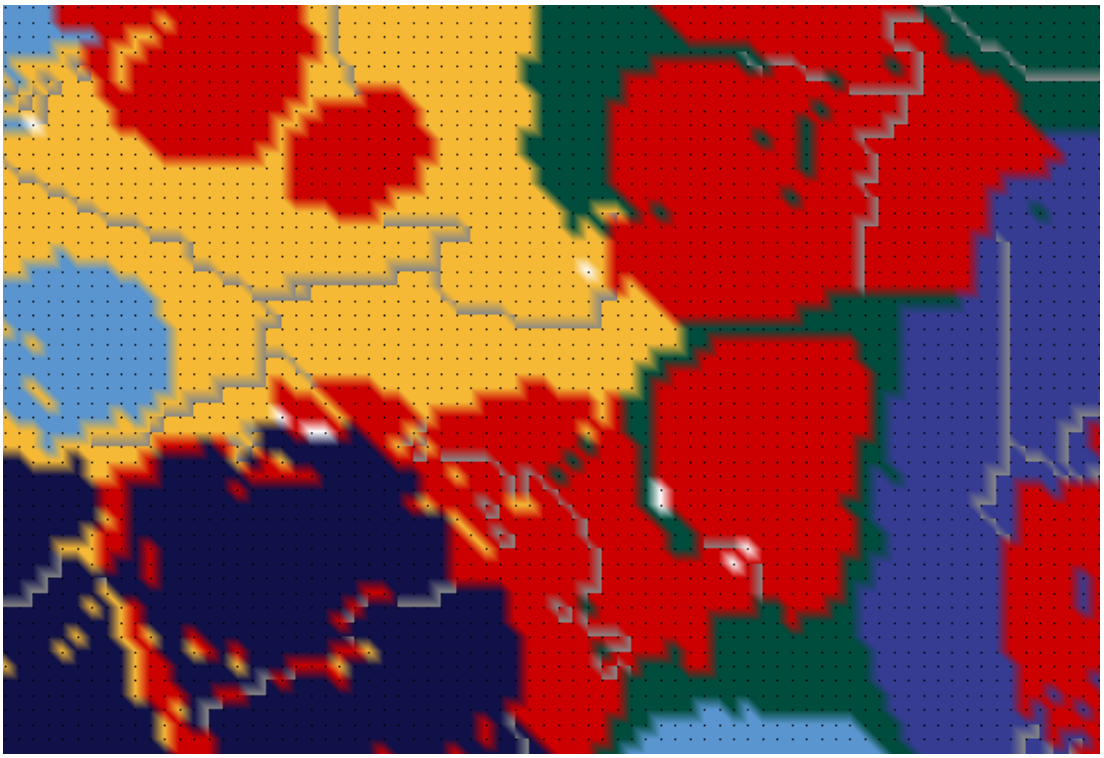
Kullanıcı etkileşimine gelirsek, kullanıcı teknenin istediği bir yerine farenin sol tuşuna basıp bıraktığında fare işaretçisinin isabet ettiği piksel pozisyonu otomatik tespit edilir. Bu pozisyon tam olarak bir düğümün üzerine denk gelirse, bu düğümü Şekil 4.2'te görüldüğü üzere dört komşu düğümün sol üst köşedeki düğümü olarak düşünüp bu dörtlünün orta noktasına boya ilave edilir. Fakat orta noktada düğüm olmadığı için ilave edilecek boya dört eşit parçaya bölünüp dört düğüme eklenir. Burada aslında bilineer interpolasyon [23] tekniğiyle madde komşu dört düğüme dağıtılmaktadır. Tekneye yeni madde ilavesinde orta noktayı tercih ettiğimiz için interpolasyonun neticesi dört eşit parça olarak gerçekleşiyor. Eare işaretçisinin isabet ettiği piksel pozisyonu dört düğüm arasında bir yere denk geliyorsa, orta nokta hesaplanıp yukarıda anlatıldığı şekilde dört düğüme eşit dağıtılır.

4.1.3 Ekrana Görüntü Çizdirme Yöntemi

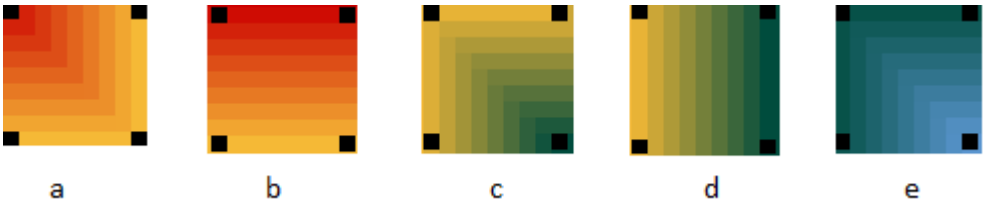
Ekrana görüntü çizdirmek, görününtünün karmaşıklığına bağlı olarak performans gerektiren bir durumdur. Görüntü kalitesini artırdıkça performanstan kaybederiz. Ebru yapım aşamasındayken kaliteli görüntü yerine daha az çözünürlüklü bir görüntü çizdirmeyi performans kaybı endişesiyle tercih ettik. 36 çözünürlüklü bir teknenin ekran görüntüsünden kesit Şekil 4.3'te görülmektedir. Normalde düğüm noktalarını çizdirmediğimiz halde çizdirme yöntemini göstermek için onları bu resim için eklenmiştir.

Çizdirme işleminde Directx'in [19] hazır bir fonksiyonunu kullandık. Buna göre, her dört düğümün düğüm bazlı pozisyonu piksel bazlı pozisyona dönüştürülür. Her dört düğümün arasında kalan piksellerin rengi bu düğümlerdeki renk değerlerinin interpolasyonu ile bulunur. Directx [19] bunu iki üçgen çizerek otomatik yapar. Şekil 4.3'teki ekran görüntüsünü %800 oranında yakınlaştırarak elde edilen farklı renkli damlalar arasındaki geçişi gösteren beş adet örnek çizim Şekil 4.4'de görülebilir.

Şekil 4.4'deki kareler, aslında iki üçgenden oluşmaktadır. Bu şekildeki (a) numaralı karenin nasıl oluşturulduğu Şekil 4.5'de görülebilir. Önce sağ üst üçgen çizdirilir. Sonra sol alt üçgen çizdirilir. Böylece kare tamamlanmış olur.



Şekil 4.3 : 36 çözünürlükte, yazılımdan alınmış gerçek bir ekran çıktısı. Düğüm pozisyonları siyah renkte düzenli noktalar halinde görülmektedir. Boyalar ise düğümlere bağlı olarak çizdirilmiştir. Aynı renkteki komşu damlalar arasına gri bir sınır çizdirilmiştir.



Şekil 4.4 : Dört düğümün arasındaki piksellerin renkleri bu düğümlerin renklerinin interpolasyonu ile bulunur.

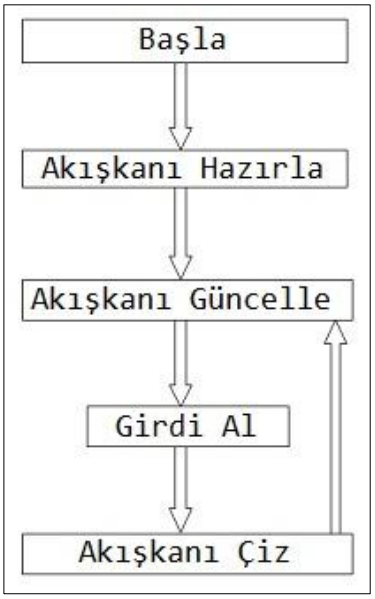
Şekil 4.5 : Bir kare çizdirilirken önce sağ üst üçgen, sonra sol alt üçgen çizdirilerek kare tamamlanır.

4.2 Programın Genel Yapısı

Programın çok genel bir işleyişi Şekil 4.6'da görülebilir. Program başlatılmasının hemen ardından akışkanın oluşturulması ve ilk değerlerinin atanması yapılır. Bundan sonra programdan çıkış butonuna basılıncaya kadar döngü halinde önce akışkanın durumu güncellenir, sonra kullanıcının sisteme yapacağı girdi işlenir, sonra akışkan ekrana çizdirilir. Ve tekrar güncelleme adımına geçilir. Ekrana çizdirme yöntemini daha önce anlatmıştık. Kullanıcı girdisi almayı hemen sonraki başlıkta inceleyeceğiz. Güncelleme ise tezimizde yaptığımız asıl katkının anlatıldığı kısımdır.

Şekil 4.6'daki "Girdi Al" başlıklı adımı kısaca anlatalım. Kullanıcının yapabildiği en temel işlem teknenin herhangi bir yerinden tekneye boya maddesi eklemesidir. Bunun yanında istediği rengi seçebilir. Boya maddesi ekleme iki şekilde olur. Birincisi, gerçek ebruda kullanılan at kılı fırçaların bir taklididir. Teknede fare ile tıkladığı koordinat merkezli belli bir alan içinde otomatik olarak rast gele damlalar oluşturulur. Damlalar, rast gele belirlenen koordinatlarda her bir damla için dörder düğüm seçilip bu damlaya işaret ettirildikten sonra bu dört düğüme boya kütlesi eklenir. Sonrasında damlanın yayılması ve diğer damlaları itmesi gibi işlemler güncelleme adımında gerçekleştirilir. İkincisi ise teknedeki boyayı karıştırmaya yarayan biz aleti ile boya maddesi eklemeyi taklit eder. Biz aleti, boya kavanozuna batırıldıktan sonra tekneye batırılır. Bizdeki boya tekneye belli bir hızda akarak yayılır. Farenin sol tuşu herhangi bir yerde basılı tutulduğunda, önce bir damla oluşturulur. Sonra oradaki dört düğüm bu damlaya bağlanır. Fare basılı olduğu süre boyunca bu dört düğüme boya kütlesi eklenmeye devam edilir. Güncelleme adımlarında bu kütle etrafa yayılmaya başlayacaktır.

Kullanıcın madde ekleyerek yaptığı etkiyi (3.15) eşitliğindeki denklemin en sağdaki teriminin bir uygulaması olarak düşünebiliriz.



Şekil 4.6 : Programın en genel çalışma akışı.

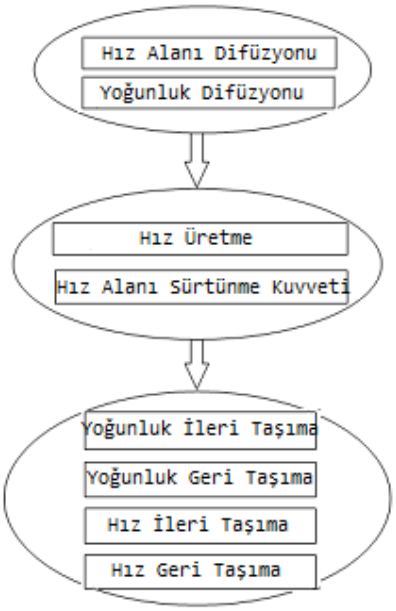
4.3 Akışkanın Güncellenmesi

Akışkanın güncelleme adımı aynı zamanda simülasyonun motoru olarak da görülebilir. Teknedeki tüm hareketi ve değerleri bu adımda yeniden düzenleriz. Güncelleme adımını üç ana kısma ayırabiliriz. Birincisinde düğümler arası madde difüzyonu veya geçişini gerçekleştiririz. İkincisinde hız vektörlerini oluştururuz. Bunun iki farklı yöntemi uygulanmıştır. Birinde düğümler arasındaki madde farklarından yola çıkarak basınç tabanlı bir hız üretim mekanizması geliştirilmiştir. Diğerinde, Lu ve diğ. [10] yayınında anlatılan matematiksel algoritma uygulanmıştır. Üçüncüsünde ise hız vektörleri boyunca hem yoğunluğu hem de hız vektörleri aktarılmıştır.

Güncelleme adımının alt kısımlarını ve onların ayrıntısını Şekil 4.7'de görebiliriz. en üstteki aşamada düğümler arasındaki yoğunluk ve hız değerlerinin birbirine geçişi hesaplanır. Bu adım, (3.15) eşitliğindeki ve (3.16) eşitliğindeki sağ taraftaki ilk terimin basit bir uygulaması olarak düşünülebilir.

Ortadaki aşamada düğümlere hız eklemesi yapılır. Bu hızların sonsuza kadar yaşamayıp zamanla yok olması için bir de sürtünme kuvveti uygulanır [20].

Son aşamada ise yoğunluklar ve hızlar, hız vektörleri boyunca önce mevcut düğümden ileriye taşınır [4]. Buna "İleri Taşıma" denir. Sonra aynı hız vektörünün tersi alınarak geriden buraya taşıma yapılır. Buna da "Geri Taşıma" denir [20]. Eşitlik (3.17) ve (3.18)'deki denklemlerin her ikisi için eşitliğin sağ tarafındaki en soldaki terimin basit bir uygulaması olarak düşünebiliriz [4,20]. Ayrıntısı sonraki başlıklarda anlatılacaktır.



Şekil 4.7 : Güncelleme adımının ana şeması.

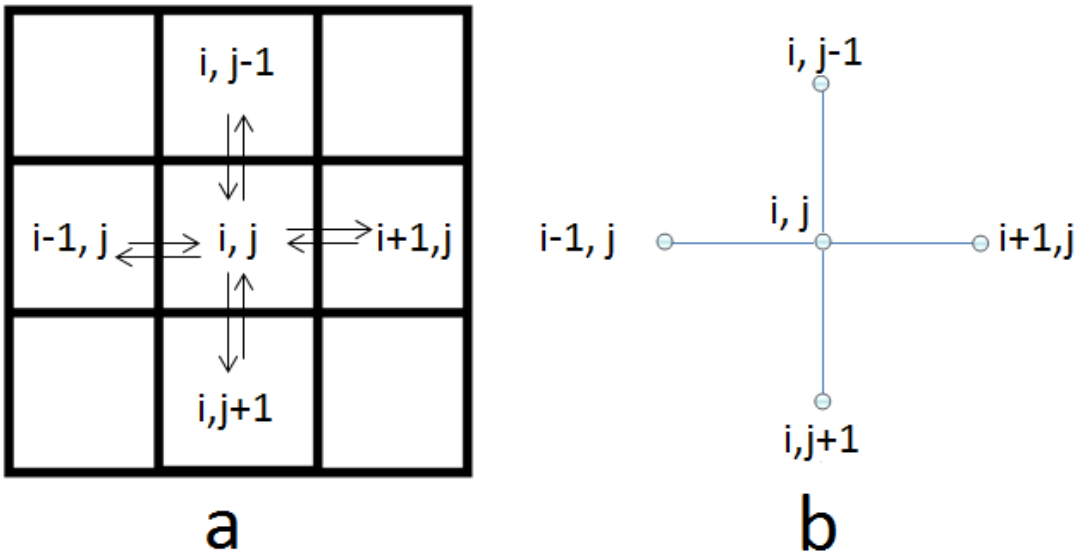
4.3.1 Yoğunluk Taşınması Kuralları

Şekil 4.7'de bulunan en üst ve en alttaki aşamalar burada anlatılacaktır. Yoğunluk değeri negatif olmayan tek bir değerden oluşur. Hız değerleri ise *x*(yatay) ve *y*(düşey) bileşeni olmak üzere negatif, sıfır ve pozitif değerler alabilir. Hız vektörlerinde işlem yaparken *x* ve *y* bileşeni birbirinden bağımsız düşünülerek algoritmalar tarafından kullanılır. Yani, mesela "Yoğunluk Difüzyonu" yapılırken yoğunluk değerlerini tutan iki boyutlu dizi parametre olarak algoritmaya verilir. Algoritma bu dizideki değerleri değiştirir. "Hız Alanı Difüzyonu" yapılırken önce hız vektörlerinin *x* bileşenlerini tutan iki boyutlu dizi parametre olarak algoritmaya verilir. Algoritma, bu dizideki değerleri değiştirir. Sonra *y* bileşenlerini tutan dizi parametre olarak verilir. İleri taşıma algoritmalarında da benzer şekilde davranılır.

Yalnız, algoritmalar yoğunluk değerlerine uygulanırken hız değerlerinden biraz farklı şekilde muamele edilir. Ayrıntılı açıklama bu kısmın alt başlıklarında verilecektir.

4.3.1.1 Difüzyon

Hız vektörlerinin difüzyonu anlaşılırsa yoğunluk değerlerinin difüzyonunu büyük ölçüde anlaşılmış olacaktır. Daha önce söylediğimiz gibi, hız vektörlerinin yatay ve düşey bileşenlerini ayrı ayrı işleme alıyoruz. Şimdi *x* yatay bileşeni üzerinden algoritmayı anlayalım. Şekil 4.8'de difüzyon, sadece dört komşu ile alış-verişin yapıldığı tipte düzenlenmiştir. Bu komşular alt, üst, sağ ve sol komşulardır. Merkezdeki düğümün indeksini (i,j) olarak aldığımızda alt komşu (i, j+1), üst komşu (i, j-1), sağ komşu(i-1, j), sol komşu (i+1,j) indekslerini alır. i sütun indeksi, j ise satır indeksidir. Başlangıç yerimiz sol üst köşedir.



Şekil 4.8 : Difüzyonun sadece dört komşu arasında olan uygulaması. a) genel gösterim. b) düğümleri ızgaranın kesişim yerlerinde düşündüğümüz gösterim.

Daha önceki bölümlerde difüzyon hakkında, her düğümden diğer komşu düğümlere bir madde akışı olur demiştik. Burada her bir düğümden dört komşusuna alış-veriş olur dedik. Bunu uygularken önce yoğunluk dizisinin bir yedeğini oluştururuz. Yoğunluk dizisini yedeğine aynen kopyalarız. Yoğunluk dizisine d_orijinal diyelim, yedeğine de d_yedek diyelim. Bunlar iki boyutlu dizidir. Yani bir nevi matrise benzer. Her bir elamanına i ve j indeks değerleriyle ulaşırız. d_orijinal[i][j] ifadesiyle dizinin istenen elemanının değerini okuyup yazabiliriz.

Her bir düğümden komşulara ve komşulardan o düğüme giden kütle miktarını her bir düğümdeki kütleyi belli bir katsayıyla çarparak elde edebiliriz [4]. Bu katsayı tüm düğümler için ortak aynı değer olabilir. Bu katsayıya difüzyon oranı manasında *k* diyelim. *k* katsayısının değeri Dt ile bir sayının çarpımıyla elde edilir. Bu sayı ise deneysel olarak belirlenmiştir. Çalışmakta olan uygulamamızda, 1 olarak alınmıştır. Dolayısıyla *k*, Dt zaman adımı değerine eşit olur. *k* büyüdükçe difüzyonda düğümler arasında gidip gelen kütle oranı artacaktır. *k*'nın değeri 1.0 olduğunu farz etsek, düğümdeki tüm madde diğer düğümlere gidecek, aynı zamanda dört komşudaki tüm kütle de buraya gelecek demektir. Bu da kütlenin uç değerler arasında salınım yapmasına sebep olur. Yani bir düğümde ya sıfır ya da dört komşusu ile kendisinin kütlesine çok yakın miktarda kütle olacaktır.

Yukarıda anlatılanlar, aşağıda (4.1) eşitliğinde verilmiştir. Burada d_orijinal d_o, d_yedek ise d_y olarak ifade edilmiştir.

```
d_y(i,j)=d_o(i,j)+k*(d_o(i,j-1)+d_o(i,j+1)+d_o(i-1,j)+d_o(i+1,j)-4*d_o(i,j))   (4.1)
```

(4.1) eşitliğindeki ifade kenardaki ve köşedeki düğümler için geçerli değildir. Mesela yatay indisi 0 olan düğümler sol kenardaki düğümlerdir. Bunların sol tarafında düğüm olmadığı için sadece üst, alt, ve sağ komşuyla alış-veriş yaparlar. Bu duruma göre düzenleme yapılırsa, (4.2) eşitliğindeki sol kenar düğümlerini de içine alan ifade elde edilir.

```
d_y(i,j)=d_o(i,j)+k *(d_o(i,j-1)+d_o(i,j+1)+d_o(i+1,j)-3* d_o(i,j))          (4.2)
```

Sağ kenar, üst kenar ve alt kenar için de benzer ifadeler sırasıyla (4.3) , (4.4) ve (4.5) eşitliklerinde verilmiştir.

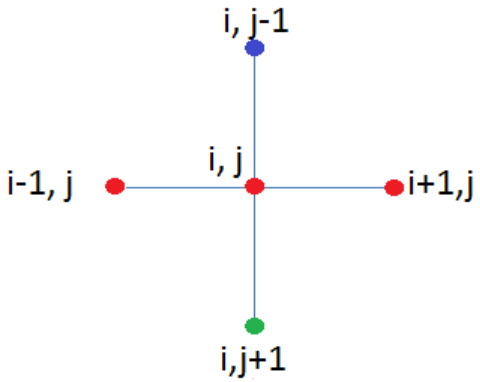
```
d_y(i,j)=d_o(i,j)+k *(d_o(i,j-1)+d_o(i,j+1)+d_o(i-1,j)-3* d_o(i,j))          (4.3)
d_y(i,j)=d_o(i,j)+k *(d_o(i-1,j)+d_o(i,j-1)+d_o(i+1,j)-3* d_o(i,j))          (4.4)
d_y(i,j)=d_o(i,j)+k *(d_o(i-1,j)+d_o(i,j+1)+d_o(i+1,j)-3* d_o(i,j))          (4.5)
```

Köşedeki düğümlerin ise sadece iki komşusu olduğu için bu iki komşuya göre işlem yapılır. Köşeler: sol üst köşe, sağ üst köşe, sol alt köşe, sağ alt köşe. Birbirine benzediği için sadece sol üst köşe için (4.6) eşitliğindeki gibi olur.

```
d_y(i,j)=d_o(i,j)+k *(d_o(i,j+1)+d_o(i+1,j)-2* d_o(i,j))                    (4.6)
```

Hız vektörlerinin *x* ve *y* bileşenlerini ayrı diziler halinde tutuyorduk. Difüzyon algoritmasını *x* ve *y* için ayrı ayrı çağırmamız gerekir. Her çağırmadan sonra d_yedek ile d_orijinal yer değiştirilir. Yani d_yedek dizisindeki değişen değer d_orijinal'e kopyalanır. Her güncelleme aşamasında süreç yeniden işler.

Yoğunluk alanının difüzyonu hız alanının difüzyonuna formül olarak çok benzemektedir. Tek fark, yoğunluk alanında aynı damlaya ait düğümler yalnız kendi aralarında difüzyon yapabilirler. Mesela bir düğümün tüm komşuları bu düğümle aynı damlaya aitse (4.1) eşitliği kullanılır. Eğer herhangi bir komşu farklı düğüme aitse o düğüm formülden çıkartılarak işlem yapılır. Hız difüzyonundaki kenar ve köşe formüllerine benziyor biraz. Şekil 4.9'daki senaryoda ortadaki düğüm, sol ve sağ taraftaki komşusuyla aynı damlaya aittir. O halde sadece sağ ve sol komşu formülde yer bulur.

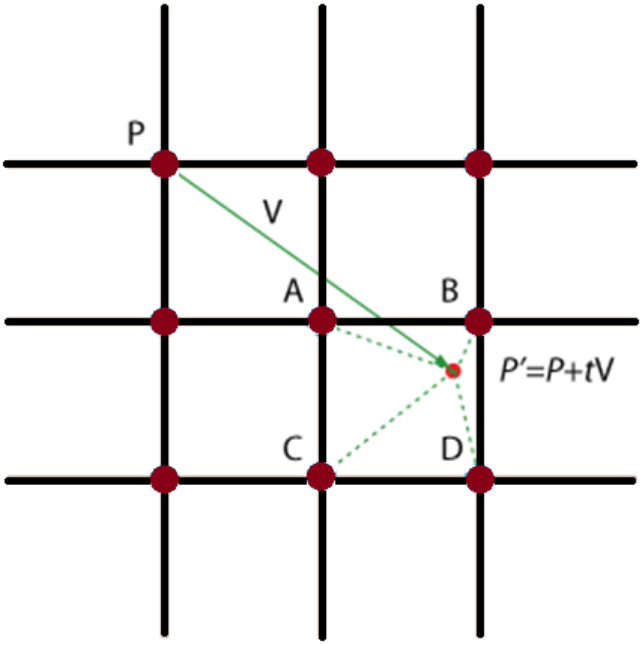



```
d_y(i,j)=d_o(i,j)+k *(d_o(i-1,j)+d_o(i+1,j)-2* d_o(i,j))
```

Şekil 4.9 : Ortadaki düğümün difüzyon formülüdür. Sadece sol ve sağ komşusu var.

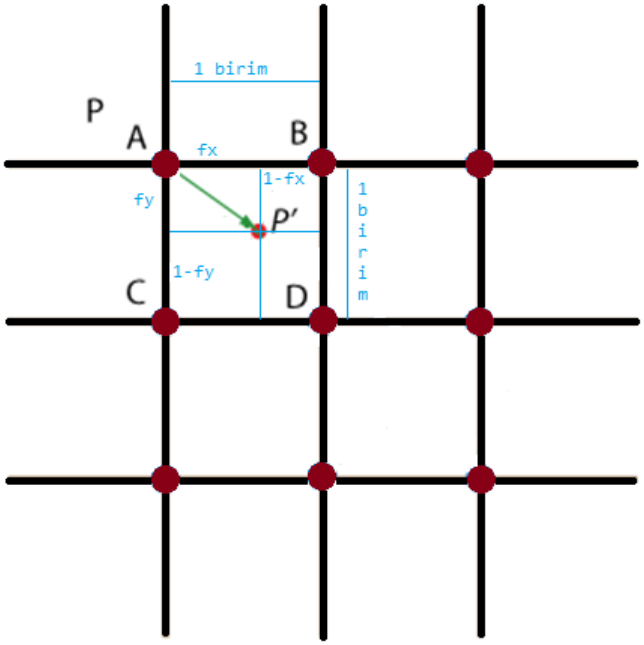
4.3.1.2 Taşıma

Taşıma adımı literatürde genellikle "advection" olarak isimlendirilir. Taşımadan kastımız, düğümlerdeki değerlerin hız vektörleri boyunca diğer düğümlere aktarılmasıdır. Stam [4]'in ileri taşıma algoritmasına ek olarak West [20]'in geri taşıma algoritmasını uyguladık. Taşıma algoritmasını hız vektörlerine uygulamak yoğunluğa uygulamaktan daha kolaydır. Çünkü yoğunluk söz konusu olduğunda farklı damlalara ait düğümler arasındaki etkileşim de dikkate alınmalıdır. Önce ileri ve geri taşıma yöntemini hız vektörlerinin *x* bileşenleri üzerinden anlatalım. *y* bileşenleri zaten *x* ile aynı şekilde işlenir. Sonra da yoğunluk değerlerindeki uygulama farkını görelim.



Şekil 4.10 : Hız vektörünün uç noktasının ve uç noktanın çevresindeki düğümlerin gösterimi.

Şekil 4.10'da tekneden bir kesit görüyoruz. P düğümü V vektörüne sahiptir. V vektörünün ucunda P' noktası bulunmaktadır. P düğümündeki miktar P' noktasına taşınmalıdır. Fakat P' noktası bir düğüm üzerine isabet etmediği için hangi düğüm ve düğümlere ne oranda pay edilmesi gerektiği belirlenmelidir. Diğer bir problem, P noktasının taşınacağı A, B, C ve D düğümleri P noktasının yan komşuları olmayabilir. Bu durumda taşınacak miktar aradaki düğümleri atlayıp geçecektir. Bu gerçek ebru fiziğinde olmayan bir şeydir. Yani her bir birim hacimdeki madde yan komşu birim hacimlere taşınabilir. Bir anda onlar etrafında yokmuş gibi uzaklara sıçrama yapmazlar. Bu durumu Şekil 4.10'daki P'= P + *t*V eşitliğindeki *t* katsayısının uygun bir rakam seçilmesiyle çözebiliriz. Bir nevi V vektörünün doğrultusunu değiştirmeden uzunluğunu azaltıyoruz. Böylece V vektörünün ucu P noktasının hemen yanı başındaki komşuları arasında bir yere denk gelir. Şekil 4.11'de bu durum gözlenebilir. *t* katsayısını difüzyon adımındakine benzer şekilde Dt ile bir sayının çarpımıyla bulabiliriz. Bu sayı deneysel bir sayıdır. Gerçek uygulamada 1 olarak aldık. Dolayısıyla *t* katsayısı Dt'ye eşit oluyor.



Şekil 4.11 : Vektör boyunca ileri taşımanın nasıl yapıldığının ayrıntılı gösterimi.

Şekil 4.11'de P noktası aynı zamanda A noktasıdır. Yani P noktasından çıkan kütlenin bir kısmı yine kendisine geri dönüyor demektir. Eğer V vektörü D noktasından çıkıyor olsaydı bu kez tüm kütle D'den çıkıp bir kısmı yine D'ye dönecekti. V vektörünün kendisinden çıktığı düğümdeki kütle bilineer interpolasyon [23] yöntemiyle A,B,C,D düğümlerine dağıtılır. Bilineer interpolasyon [23], lineer interpolasyondan yola çıkılarak hesaplanır. Bilineer interpolasyonun [23] ismi lineer olmasına rağmen lineer değildir. İçinde "*xy*" çarpımı terimlerini barındırır. Şekil 4.11'de görüldüğü üzere A,B,C,D düğümlerinin oluşturduğu karenin bir kenar uzunluğu 1 birim alınır. Böylece bilineer interpolasyon [23] hesabı basitleşir. Her bir düğüme eklenecek kütle miktarı Çizelge 4.2'de görülebilir. A'daki kütleye M dersek bu kütle önce A'dan çıkartılır. Sonra bilineer interpolasyona [23] göre ilgili düğümlere dağıtılır. Dağıtılan miktarlar toplandığında M'ye eşit çıktığı görülecektir. Buradan taşıma yönteminin toplam kütle miktarını koruduğu anlaşılır. İşlem her bir düğüm için tekrarlanır. Difüzyonda olduğu gibi bir orijinal bir de yedek dizi tutulur. Orijinaldeki bilgi, yedeğe kopyalanır. Çıkarılacak miktar orijinalden çıkarılır. Ekleme yedeğe eklenir. Tüm düğümler işlemden geçtikten sonra yedekteki bilgi orijinalle yer değiştirilir.

Çizelge 4.2 : İleri taşıma algoritmasında her bir düğüme ne kadar kütle gideceği bilineer interpolasyonla belirlenir. A, B, C veya D'den birindeki taşınmaya aday kütleye M diyebiliriz. Burada A düğümüdür.

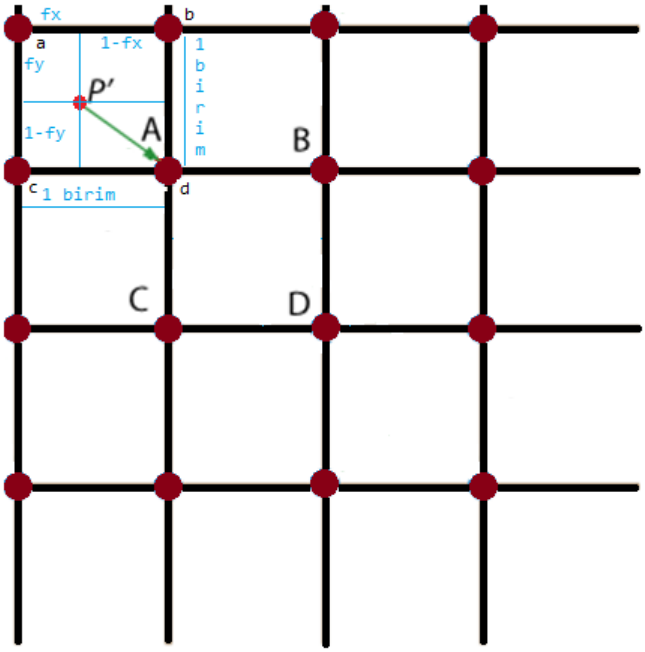
Düğümler	Eklenecek Miktar	Çıkarılacak Miktar
A	(1-fy) * (1-fx) * M	M
B	(1-fy) * (fx) * M	M
C	(fy) * (1-fx) * M	M
D	(fy) * (fx) * M	M

Şimdi geri taşıma nasıl yapılıyor görelim. Aslında geriye taşınmıyor. Gerideki hız değerleri kısmen hazır düğüme taşınıyor. Geri taşıma algoritmasını anlamayı biraz daha kolaylaştırmak için Şekil 4.12'den faydalanabiliriz. Dt değeri pozitif bir sayıdır. V vektörünü Dt'nin negatifi yani –Dt ile çarptığımızda V vektörünün bu sefer *x* ve *y* eksenlerine göre tersini almış oluruz. Vektörün orijin noktasıyla uç noktasını yer değiştirdiğimizi farz etsek Şekil 4.12'deki P' noktasını elde ederiz. P' noktasının çevresindeki düğümlerden çıkacak miktarlar bilineer interpolasyonla [23] bulunur. P' noktasının sol üst köşesindeki düğüm a, sağ üst köşesindeki b, sol alt köşedeki c, sağ alt köşe d düğümü olsun. Her bir düğümden çıkacak miktar o düğümdeki mevcut miktarın bir sayıyla çarpılmasıyla bulunur.

Çizelge 4.3'te düğümlerden çıkan ve giren miktarlar belirtilmiştir. İleri taşımada olduğu gibi iki dizi ile işlem gerçekleştirilir. d_orijinal dizisi şu andaki mevcut miktarları tutan dizidir. d_yedek işlem sonunda oluşacak yeni durumu tutan dizidir. İşlem başında d_orijinal d_yedeğe aynen kopyalanır. Ma, Mb, Mc, Md miktarları d_orijinal dizisinden elde edilir. Ekleme ve çıkarma d_yedek dizinde yapılır. Tüm düğümler işlemden geçtikten sonra d_yedek dizisi d_orijinal dizisine aynen kopyalanır. Hız bileşenlerinin y bileşeni içinde aynı işlem tekrarlanır.

Çizelge 4.3 : Geri taşımada herbir düğümden çıkan miktar bilineer interpolasyonla bulunur. Ma, Mb, Mc, Md her bir düğümdeki mevcut miktarı gösterir.

Düğümler	Eklenecek Miktar	Çıkarılacak Miktar
a	0	(1-fy) * (1-fx) * Ma
b	0	(1-fy) * (fx) * Mb
c	0	(fy) * (1-fx) * Mc
d	Sağ taraftaki çıkarılacakların toplamı	(fy) * (fx) * Md

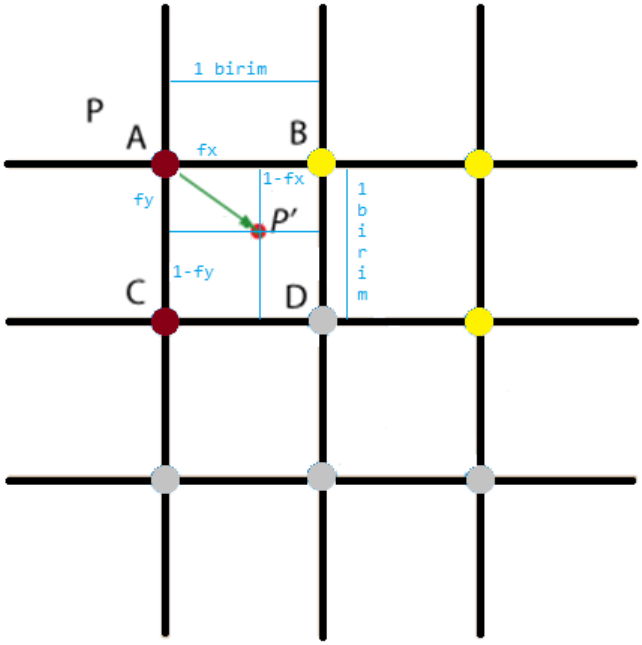


Şekil 4.12 : Geri taşıma algoritmasının çalışması. A düğümünden çıkan vektör –Dt ile çarpılıp şekildeki gibi düzenlenir. P' noktasının etrafındaki dört düğümden bilineer interplosyanla bir miktar kütle A noktasına taşınır.

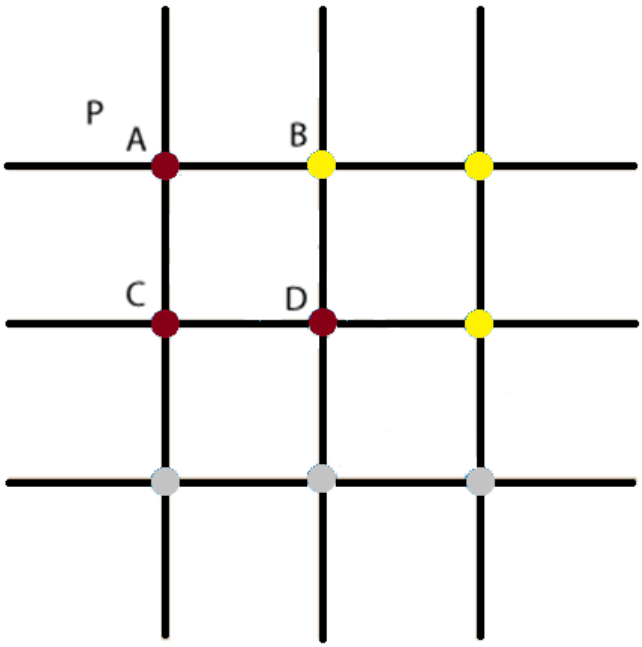
Hız için olan ileri ve geri taşıma algoritmasını yukarıda anlattık. Boya kütlesi taşıma algoritması büyük oranda bunlara benzemekle birlikte kısaca söylersek farklı damlalara ait düğümler arasında madde taşıması yapılmaz. Önce ileri taşıma algoritmasını anlatalım.

Hız vektörlerindeki ileri taşımada olduğu gibi bilineer interpolasyonla [23] dört düğüme dağılacak miktar aynı şekilde hesap edilir. A, B, C, D düğümlerinin her biri için bakılır. Bu düğüm eğer kaynak düğümle aynı damlaya aitse payına düşen miktar eklenir. Eğer boş düğümse ve damlanın mevcut alanı ideal alanından küçükse payına düşen miktar aynen eklenir, damlanın alanı bir artırılır ve düğüm bu damlaya bağlanır. Eğer başka bir düğümse ve içindeki madde miktarı sıfır ise ve damlanın mevcut alanı ideal alanından küçükse payına düşen miktar aynen eklenir, damlanın alanı bir artırılır ve düğüm bu damlaya bağlanır. Eğer başka bir düğümse ve içindeki madde miktarı sıfır değilse bu pay dağıtılmaz. Kaynak düğümde bırakılır.

Şekil 4.13'te B düğümü sarı renkli farklı bir damlaya aittir ve içinde madde vardır. C düğümü kaynak düğümle aynı damlaya ait. D düğümü de boş düğüm olsun. İleri taşıma gerçekleştiğinde Şekil 4.14'teki gibi bir durum oluşur. D düğümü, A düğümünün ait olduğu damlaya bağlanır. Bir düğümden ileri taşıma yapılabilmesi için o düğümdeki madde miktarı damlanın öd değerinden büyük veya büyük olmalıdır. Damlanın kütlesinin ideal alanına bölümüyle öd değerini elde ediyorduk. Öd aynı zamanda özkütle manasını da barındırır. Bir düğümden ileri taşıma yapılırken o düğümün kendisine de madde eklendiği için. O düğümdeki madde hiçbir zaman tükenmeyecektir. Bu durumda başka damlalar o düğüme taşınamayacaktır. İleri taşıma yapacağımız bir düğümdeki madde miktarı belli bir rakamın altına düşerse bu düğüme düşen taşıma payı diğer düğümlere eşit paylaştırılır. Bu düğüm boşaltılır.



Şekil 4.13 : A kaynak düğümdür. B farklı bir damlaya ait içinde kütle bulunan bir düğümdür. D boş düğümdür. Yani hiç bir damlaya ait değildir.



Şekil 4.14 : Şekil 4.13'deki A noktasından ileri taşıma gerçekleştiğinde D düğümü, A düğümünün damlasına bağlanır. B değişmez. A'dan çıkan miktar, A, C ve D düğümlerine paylaştırılır.

Kütlenin geri taşınması, hızın geri taşınmasından biraz farklıdır. Hız değerlerimiz negatif olabiliyordu. Kütle değerleri ise negatif olamaz. İleri taşımada bir düğümden belirli dört düğüme kütle aktarılırken, geri taşımada dört düğümden bir düğüme aktarım yapılıyor. Fakat bu dört düğüm aynı zamanda başka düğümlere de kütle gönderiyor olabilir. Dört düğümden her bir düğümdeki mevcut kütleden daha fazla kütle dışarıya gönderilemeyeceğine göre, her birinden çıkan toplam miktar, gönderilecekleri düğümlere pay edilmelidir.

Geri taşımada gerideki düğümlere kaynak düğümler, bunların kütle gönderdiği düğümlere hedef düğüm deriz. Hedef düğümler için bunların kaynak düğümlerini tutmak üzere düğüm sayısıyla aynı boyutta bir dizi tutulur. Bu dizideki her bir eleman, dört kaynak bilgisini sol üst köşedeki düğümü kaydederek tutmuş olur. Başka bir dizide, her bir hedef düğümün kaynaklarının her biri için bilineer interpolasyon [23] oranı tutulur. Dolayısıyla bu dizinin boyutu düğümler adetinin dört katıdır. Bir başka dizi de her bir kaynak düğüm için o düğümden çıkan bilineer interpolasyon [23] değerleri toplamını tutar. Boyutu düğümler adeti kadardır. Tüm düğümler için hesap yapıldıktan sonra kütle aktarımına başlanır. Bir hedef düğümün her bir kaynak düğümünden gelecek miktar o kaynaktaki kütle miktarının kaynaktan hedef düğüme giden oranın kaynaktan çıkan toplam orana bölümüyle çarpılarak bulunur. Kaynaklardan çıkarılacak miktar, yedek diziden çıkarılır. Hedef düğüme eklenecek miktar yedek diziye eklenir. Tüm düğümler işlendikten sonra yedek dizi orijinal diziye kopyalanır. Önemli bir nokta ise hedef düğümden farklı damlaya ait kaynak düğümlerden kütle alınmaz. Onlar yok farz edilir. Bir başka nokta ise eğer bir kaynak düğümden çıkan oranlar toplamı birden küçükse bire tamamlanıp öyle işleme alınır. Böylece o kaynaktaki tüm kütlenin taşınması engellenir. Mesela bir kaynak düğümden iki hedefe gönderim olacak diyelim. Biri 0.25, diğeri de 0.25 oranında pay isterse toplam 0.5 olur. Her biri bu kaynaktaki kütlenin yarısını alıyor demektir. Halbuki her biri dörtte bir almalıydı. 0.5'i 1'e tamamlayarak bunu garanti ediyoruz.

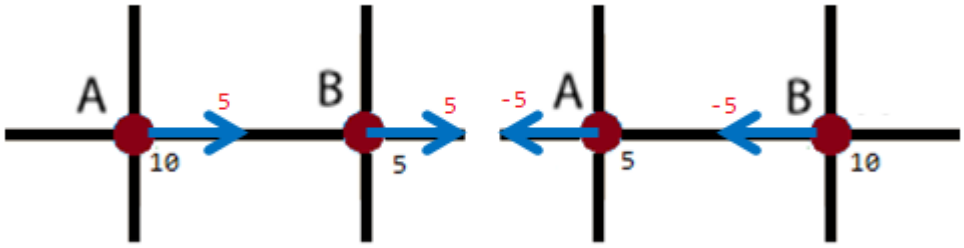
4.3.2 Hız Vektörleri Alanının Üretim Algoritması

Hız vektörlerinin üretilmesi teknedeki boya hareketinin sağlanması için olmazsa olmaz bir aşamadır. Algoritmaların çeşitlenmesinde önemli bir yere sahiptir. Bu tez çalışmasında iki çeşit hız vektörü üretim algoritması uyguladık. Basınç tabanlı algoritmada boyanın teknede dağılma mekanizmasına benzeyen bir yaklaşım uygulanmıştır. Özetlersek, boya maddesi bir akışkan olarak basıncın yüksek olduğu yerden basıncın alçak olduğu yere doğru hareket eder. Matematiksel algoritmada ise her bir damlanın etrafına matematiksel bir formülle hesaplanan bir itme kuvveti uyguladığı düşünülür. Böylece boya damlaları birbirini iterek yer değiştirmelerine sebep olur. Bu bölümün alt başlıklarında ilgili algoritmalar ayrıntısıyla anlatılacaktır.

4.3.2.1 Basınç Tabanlı Algoritma

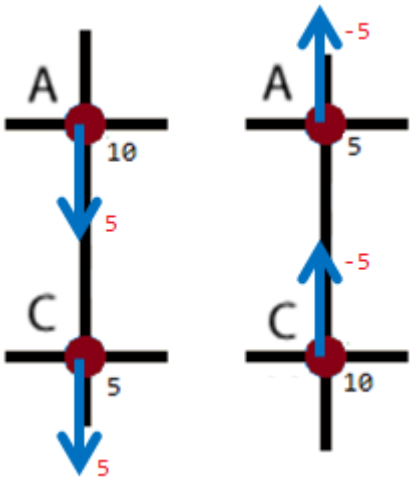
Ebrunun sayısal modelini düğümler üzerine kurmuştuk. Buna göre boya maddeleri düğümler üzerinde bulunuyor ve bir düğümden diğer düğümlere hız vektörleri boyunca taşınıyordu. Her bir düğüm üzerinde skalar bir kütle miktarı bulundurur ve bir damlaya aittir. Aynı damlaya ait düğümler arasında kütle miktarının çok olduğu düğümden az olan düğüme doğru basınç uygulanır. Bu basınç iki düğümün kütle farkları alınıp iki düğüme hız vektörü şeklinde eklenerek gerçekleştirilir. Bir düğümün sekiz adet komşusu vardır. Bunlardan sadece üst, alt, sağ, ve sol komşular alınabileceği gibi çapraz komşular da dikkate alınabilir. Biz çapraz komşuları da hesaba kattık.

Önce sağ taraftaki düğümle olan basınç ilişkisini görelim. Şekil 4.15 sol tarafta görüldüğü üzere, A düğümü ve B düğümü aynı damlaya aittir. A düğümümündeki kütle miktarı 10 birim, B düğüdeki ise 5 birimdir. A düğümündeki kütle miktarından B düğümündeki kütle miktarı çıkartılıp elde edilen fark her iki düğüme eklenir. Böylece her iki düğümde +x yönünde 5 büyüklüğünde bir hız vektörü eklenir. Şekil 4.15 sağ taraftaki gibi A düğümünde 5, B düğümünde 10 birim kütle olsaydı fark -5 olurdu. Bu sefer her iki düğümde -x yönünde 5 büyüklüğünde bir hız vektörü eklenir.



Şekil 4.15 : Aynı satırdaki iki düğüm arasındaki basınç ilişkisi.

Şekil 4.15'e dikkat edilirse B düğümü, A düğümünün sağındaki düğümdür. A düğümü ise B düğümünün solundaki düğümdür. Tek bir işlemle A düğümünün sağ taraftaki düğümle olan basınç ilişkisi ile B düğümünün sol taraftaki düğümle olan basınç ilişkisi hesaplanmış oluyor [20]. İki boyutlu dizi şeklinde tuttuğumuz düğümlerde işlem sırası sol üst köşeden başlayıp sağa doğru ilerler. İlk satır bitince ikinci satıra geçilir. Böylece sağ alt köşeye kadar işlem sürer. A düğümüne sıra geldiğinde bunun sol taraftaki düğümle olan ilişkisi çoktan hesaplanmış olacaktır. Kütle farkının her iki düğüme eklenmesi işlem açısından böyle basitlik sağlıyor. Aynı şekilde A düğümünün alt tarafındaki düğümle olan ilişkisi hesaplandığında, alt taraftaki düğüm için de üst taraftaki düğümle olan basınç ilişkisi hesaplanmış oluyor. A düğümünün sol alt çaprazdaki düğümle olan basınç ilişkisi aynı şekilde hesaplandığında, sağ üst çaprazdaki düğümle olan basınç ilişkisi de hesaplanmış olacak. Sağ alt çaprazdaki düğümle olan basınç ilişkisi hesaplandığında sol üst çaprazdakiyle olan ilişki de hesaplanmış olur. Çapraz düğümler arasında kütle farkı 0.7 ile çarpıldıktan sonra hız vektörü olarak eklenir. Çünkü uzaklık arttıkça etkinin azalması gerekir. Yatay veya düşey komşu düğmeler 1 birim uzaklıktaysa, çapraz düğümler ise yaklaşık 1.4 birim uzaklıktadır. Ters orantı alındığında yaklaşık 0.7 ile çarpılması gerektiği anlaşılır. Çapraz komşularda aynı anda hız vektörlerinin x ve y bileşenlerine ekleme yapıldığı için 0.7 ile çarpılan değer 1.4'e bölünüp x ve y bileşenlerine ayrı ayrı eklenir. Burada sadeleştirme yapıldığında 0.7 ile çarpıp 1.4'e bölmek yerine kütle farkı doğrudan 0.5 ile çarpılıp aynı sonuç elde edilebilir.

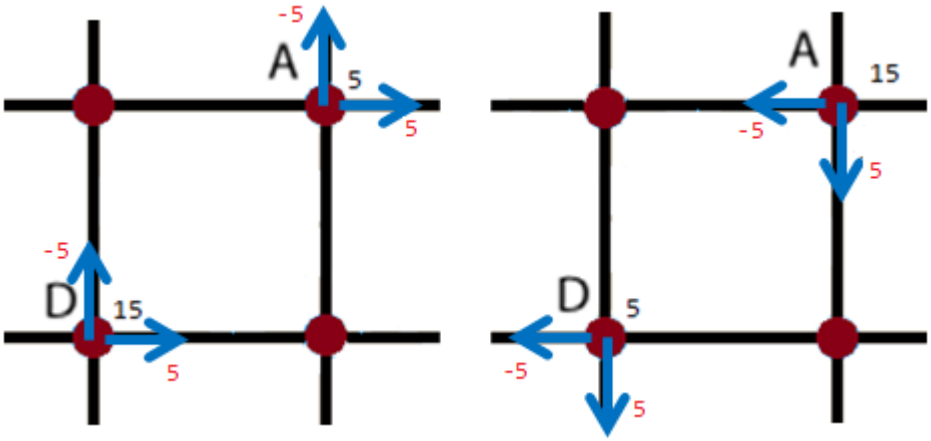


Şekil 4.16 : Üst ve alt düğümler arasındaki basınç ilişkisi

Üst ve alt komşuların basınç ilişkisinin hesaplanmasını görelim. Şekil 4.16'ya baktığımızda A düğümü ve C düğümü, sırasıyla 10 ve 5 birim kütleye sahiptir. A'dan C'yi çıkararak elde edilen 5 değeri +y yönünde her iki düğümün hız vektörüne eklenir. Yani her iki düğümün hız vektörlerinin *y* bileşenine 5 eklenir. Eğer A'nın kütlesi 5, C'ninki ise 10 olsaydı *y* bileşenlerine -5 eklenecekti.

Sol alt çapraz düğümle olan basınç ilişkisini hesaplayalım. Şekil 4.17'de A düğümü 5 birim kütleye sahip, D düğümü 15. Kütle farkı -10 yapar. Daha önce açıklandığı üzere bunu 0.5 ile çarparak elde edilen -5 değeri, *x* bileşeninden çıkarılırken, *y* değeriyle toplanır. Bu sayede Şekil 4.17 sol taraftaki kısımdaki hız vektörleri doğrultularına uygun değerler tespit edilir. Sağ çaprazı hesaplıyor olsaydık, -5 değeri *x* ve *y* ile ayrı ayrı toplanacaktı.

Eğer A düğümünde 15, D düğümünde 5 birim kütle olsaydı, yeni durum Şekil 4.17 sağ taraftaki gibi olacaktı. Sağ çapraz da buna benzer şekilde olduğu için onu anlatmayacağız. Sağ, alt, sağ alt çapraz, sol alt çapraz komşular olmak üzere dört komşuyu işlemden geçirince sol, üst, sol üst çapraz, sağ üst çapraz komşular otomatikman hesaplanmış oluyor.

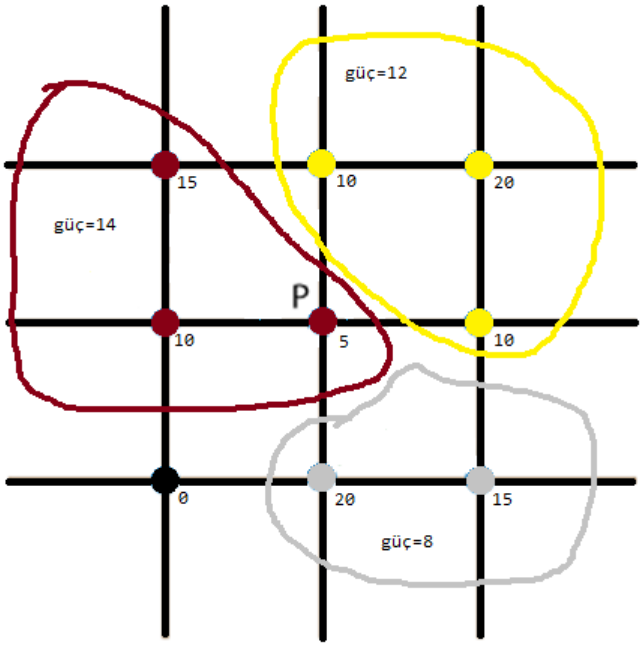


Şekil 4.17 : Sol alt çapraz komşuların basınç ilişkisi.

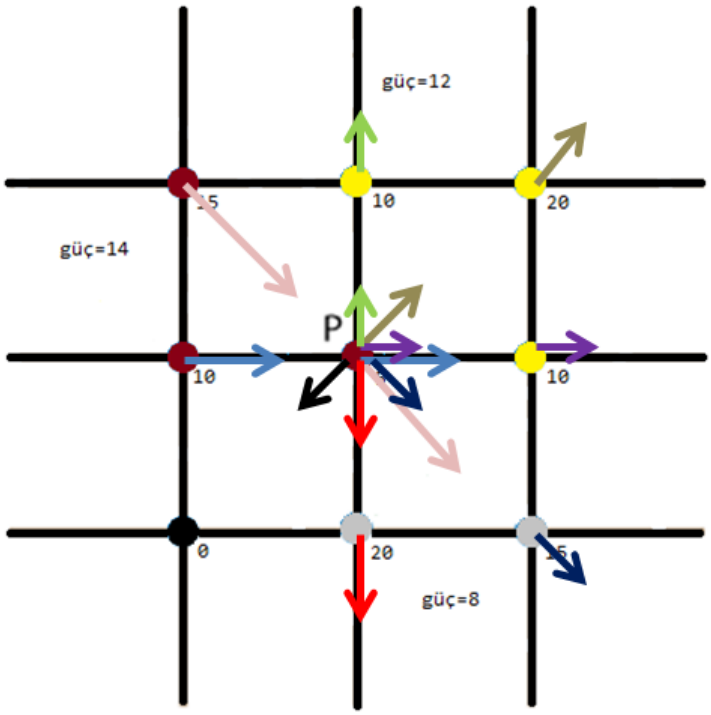
Komşu düğüm hiç bir damlaya ait değilse yani boş düğümse fark alınır ve yalnızca A düğümüne eklenir. Boş düğümlerdeki kütle miktarı sıfır olduğu için kütle farkı A düğümündeki kütle miktarına eşit olacaktır.

Eğer komşu düğüm farklı bir damlaya aitse, kütle farklarını almak yerine güç farklarını alıp devam ederiz. Bir damlanın güç değeri, toplam kütlesinin o andaki kapladığı toplam düğüm sayısına bölümüyle bulunur. Farklı damlalar arasında güç farkının alınmasını önemli bir sebebi var. Mesela iki damla düşünelim birincisinin Öd değeri 10 olsun, ikincisininki 15. İkinci damla ideal alanına ulaşmak üzere olsun. Güç değeri 15'e yakın olur. İkinci damlanın düğümlerinin her birinde yaklaşık 15 birim kütle olacaktır. Birinci damla ise henüz ideal alanının yarısına ulaşmış olsun. Bunun gücü 20 olacaktır. Burada gerçekte birinci damlanın ikinci damlayı itmesi beklenir. Birinci damla yayılma aşamasında olduğu için düğümlerindeki kütle dağılımı homojen değildir. Sınır düğümlerinde çok az kütle bulunur. İkinci damla ise yayılmasını bitirmek üzere olduğu için toplam kütle düğümlere homojen dağılır. Sınır düğümlerinde öd miktarına yakın kütle miktarı vardır. Şu halde yaklaşık 15 birim kütle olsun demiştik. Birinci damlanın sınır düğümlerinde ise 5 birim kütle olduğunu farz edelim. Şimdi kütle farkı alınırsa ikinci damlanın sınır düğümü birincininkinden yaklaşık 10 birim fark atıp baskın gelerek birinci damlayı itecektir. İstediğimiz bu değildi. Güç farkı alınsaydı, birinci damlanın gücü 20, ikinci damlanınki 15 olduğundan fark 5 olur. Şimdi birinci damla ikinciyi itip kendine yer açabilir.

Bir düğümün tüm komşularıyla olan basınç ilişkisini bir resim karesinde görmek için Şekil 4.18 ve Şekil 4.19'a dikkat edebilirsiniz. Şekil 4.18'de üç adet kırmızı düğüm kırmızı renkli damlaya, üç adet sarı düğüm sarı damlaya, iki adet gri düğüm gri damlaya aittir. Bir adet boş düğüm bulunmaktadır. Düğümlerin kütle miktarları yanlarında verilmiştir. Damlaların yaklaşık sınır çizgisi aynı renkle çizilmiş olup içine damlanın güç değeri yazılmıştır. Şekil 4.19'da mevcut durumda P düğümünün çevresine etkisi ve çevresinin P düğümüne etkisi vektörlerle gösterilmiştir. P noktasına diğer vektörlerin etkisinin bileşkesi alınarak net hız vektörü elde edilir. Aynı renkli vektörlerin büyüklük ve doğrultuları aynıdır.



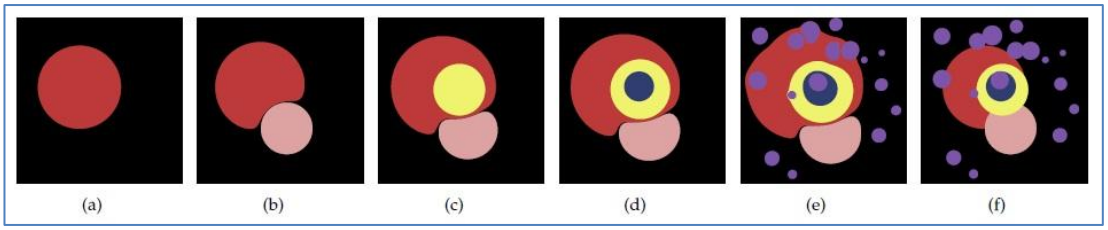
Şekil 4.18 : Bir düğümün tüm komşularıyla olan basınç ilişkisi. Siyah düğüm boş düğümdür. Kırmızı, sarı ve gri düğümler ilgili renklerdeki damlalara aittir.Damlaların sınırlarının temsili çizimi düğümlerle aynı renkte verilmiştir. Damlaların güç değerleri damla sınırları içinde, düğümlerin kütle değerleri de yanı başlarında verilmiştir.



Şekil 4.19 : Mevcut damla ve düğüm yapısına göre P düğümünün diğer düğümlerle olan basınç ilişkisi gösterilmiştir.

4.3.2.2 Matematiksel Algoritma

Lu ve diğ. [10]'nun matematiksel algoritmasının Battal Ebru için olan kısmını kısaca şöyle açıklayabiliriz. Damlalar aktif ve pasif olarak ikiye ayrılır. Aktif damla etrafa yayılmaya devam eden damladır. Yani kendisi genişlerken etrafındaki damlaları iter. Pasif damla ise yayılımını tamamlamış damladır. Algoritma gerçek zamanlı olmadığı için damla damlatılır damlatılmaz tüm işlem anında olup biter. Ardından aktif damla pasif olur. Şekil 4.20'de süreç açıkça görülebilir. Damlalar güzel bir şekilde birbirini etkiliyor. Fakat sanatçı zamana bağlı olarak göremiyor.



Şekil 4.20 : Battal Ebru için Lu'nun algoritmasının işleyişi. (a) İlk damla damlatılıyor, ve anında olması gereken boyuta geliyor. (b) Yanına bir başka damla damlatılıyor, yeni damla kendi boyutuna anında ulaşırken önceki damlayı itiyor. (d,e) yeni damlalar aynı şekilde işleme devam ediyor. Eğer damlalar birbirini itmeseydi şekil (f)'deki gibi olacaktı.

Şekil 4.20'de Lu [10]'nun Battal Ebru algoritmasının süreci görülmektedir. Soldan sağa doğru damla ekleye ekleye ve her eklenen damla diğerlerini iterek oluşan şekil, (e) numaralı resimde görülebilir. Eğer itmeseydi (f)'deki gibi olacaktı.

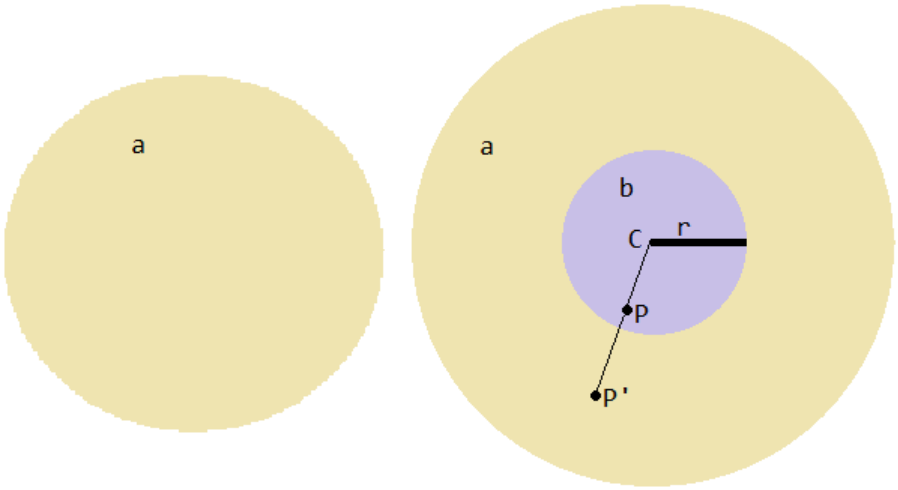
Bu algoritmanın arkasındaki basit matematiği kısaca açıklayalım. Bir damla başka bir damlanın içine düştüğünü farz edelim. Her bir damlanın ulaşacağı bir azami bir alan vardır. Bu alan dairesel düşünülürse, belirleyici unsuru yarıçapı olur. Buna R diyelim. Birinci damlanın azami alanı a, ikincisi b olsun. İkinci damla düşüp yayılma işlemi gerçekleştikten sonra toplam alan a'dan a+b'ye yükselir. Birinci damlanın merkezindeki noktalar yarıçap 0'dan $\sqrt{(b/\pi)}$ 'ye gelirken, kenardaki noktalar da $\sqrt{(a/\pi)}$'den $\sqrt{((a+b)/\pi)}$'ye gelir. Bir C merkezli düzlemde, P noktası verildiğinde bu noktanın işlem sonrası taşınacağı noktaya P' diyelim. P, P', C noktaları *x* ve *y* koordinatlarını tutan iki boyutlu birer vektördür. (4.7) eşitliğinde formül görülebilir. (4.8) eşitliğinde algoritmanın zamana bağlı olarak düzenlenmiş hali görülmektedir. Şekil 4.21'de, eşitliklerdeki terimler kısmen görülebilir.

$$\overrightarrow{P'} = \vec{C} + \left(\vec{P} - \vec{C}\right)\sqrt{1 + \frac{r^2}{|\vec{P}-\vec{C}|^2}} \qquad \textbf{(4.7)}$$

$$\overrightarrow{P'} = \vec{C} + \left(\vec{P} - \vec{C}\right)\sqrt{1 + \frac{r^2.Dt}{|\vec{P}-\vec{C}|^2.s}} \qquad \textbf{(4.8)}$$

(4.8) eşitliğinde 'dt' terimi, geçen süreyi gösterir. 's' terimi ise damlanın ömrünün saniye olarak değeridir. Gerçek ebruda damlalar ortalama 3 saniyede yayılmasını tamamladığı gözlenmiştir. 'r' terimi ise yayılmakta olan damlanın yayılmasını tamamladığında ulaşacağı azami yarıçap değerini gösterir. 'r' her bir damla oluşturulurken(damlatılırken) bir kez belirlenir. Sonradan değişmez. 's' terimi de aynı şekilde damla oluşturulurken belirlenir. Sonradan değişmez. 'dt' ise kontrol bize her geçişinde haliyle değişmektedir. Her iterasyonda tüm düğümler için aynı 'dt' değeri kullanılır. 'r' ve 's' değerleri birbirine bağımlıdır. O yüzden uygun sonuçlar için, birbirine göre durumları optimize edilmesi gereken iki önemli parametredir.

Eğer 'r' sabitken, 's' değerini artırırsak damla yavaş yayılacaktır. Yani aynı boyuta daha uzun sürede ulaşacaktır. Eğer 's' değerini azaltırsak, damla hızlı yayılacaktır. Yani azami boyutuna daha çabuk ulaşacaktır. 's' değeri sabitken 'r' değerini artırırsak, damlanın azami boyutu artacaktır. Daha büyük bir boyuta aynı sürede ulaşacağı için dağılma hızı artacaktır. Bu matematiksel algoritma bizim basınç tabanlı algoritmadan daha hızlıdır. Neredeyse iki katı hızlıdır.



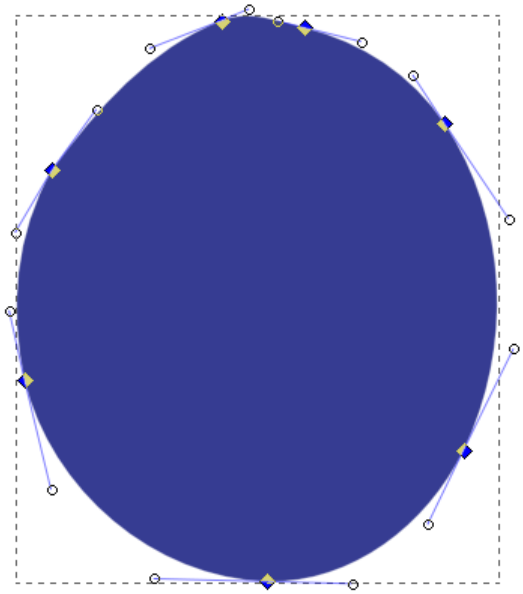
Şekil 4.21 : Soldaki açık sarı renkli damla a alanına sahip ilk damladır. Onun içine mor renkli damla damlatıldığında sarı alanın şekli değişse de alanı değişmez. Mor renkli damla C merkezinden etrafına itme kuvveti uygular. İşlem bir anda olup biter. P noktası, P' noktasına taşınır.

4.4 Vektörel Çıktının Üretilmesi

Vektörel çıktıdan kastımız SVG [21] dosya biçimidir. XML yapısını kullanarak görsel öğeleri parametrik değerlerle ifade eden bir yöntemdir. Çizelge 4.4'te içi dolu bir eğrinin nasıl kodlandığı verilmiştir. Aynı damlanın noktaları ve bu noktaların ikişer adet kontrol noktaları Şekil 4.22'de gösterilmiştir.

Çizelge 4.4 : Bir damlanın SVG biçiminde kodlanması.

```
<path stroke="#ffffff" stroke-opacity="0.31" stroke-width="0.2" fill="#f5b936"
d=" m 361.78,246.94 c -5.26,8.94 -6.11,20.04 -3.90,30.05 3.76,15.63 18.41,28.37
34.66,28.77 12.05,0.30 22.85,-8.14 27.93,-18.68 7.18,-14.61 6.47,-33.19 -2.66,-
46.80 -4.55,-6.96 -11.89,-11.71 -19.92,-13.69 -3.87,-0.92 -7.96,-2.70 -11.91,-1.01 -
10.30,3.99 -17.84,12.66 -24.18,21.38 z" />
```



Şekil 4.22 : Çizelge 4. 4'teki "path" elemanın vektörel çizimi. Kare şeklindeki noktalar ana noktaları gösterir. Bunlardan çıkan ikişer adet kontrol noktaları çizgilerin ucundaki yuvarlak noktalar ile gösterilmiştir.

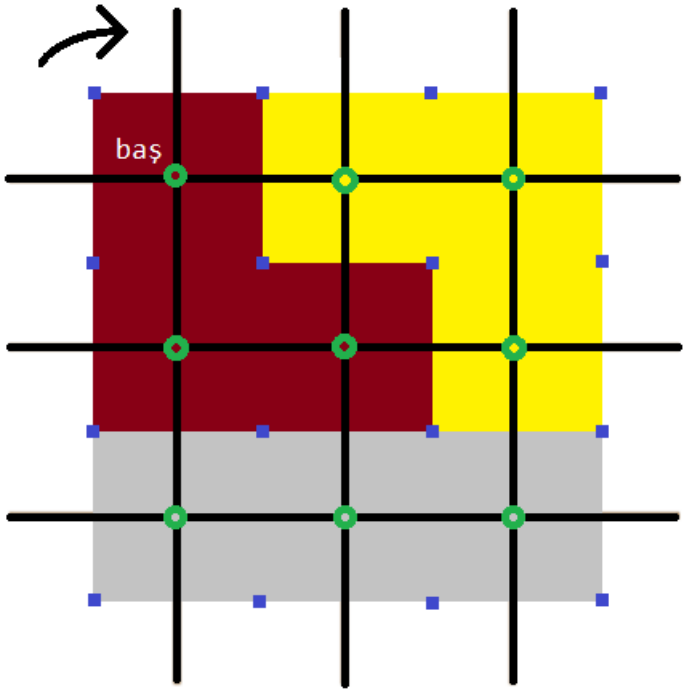
Kontrol noktaları eğrilik oranını belirliyor. Eğriliğin hesaplanmasında kullanılan yöntem, Spiro Eğrilerinde [13] kullanılaln bir çeşti otomatik yumuşatma yöntemidir. Kontrol noktalarını buna göre oluşturduğumuz için herhangi bir vektörel çizim yazılım paketi, Şekil 4.22'deki görüntüyü çizdirecektir.

Sınır noktalarının tespit edilmesi gerekiyor. Her bir düğüm bir karenin ortasında bulunuyor. Bu karenin damlanın dışına bakan köşeleri bizim asıl sınırlarımız olmalıdır. Yoksa iki komşu damla arasındaki mesafe artacaktır.

Sınır bulma algoritmasını kısaca açıklayalım. İlk önce sol üst köşeden başlayarak sağa doğru düğümler üzerinde gezeriz. İlk rastladığımız sınır düğümün ait olduğu damla, etrafını dolanacağımız damladır. Sağa doğru gezmeyi kaldığı yerde bırakırız. Bir düğümün üst, alt, sağ veya sol komşusundan en az biri başka bir damlaya aitse bu düğüm sınır düğümdür. İlk rastladığımız sınır düğüme baş düğüm diyelim. Bu düğüm bir karenin ortasında bulunuyor demiştik. Bu karenin köşeleri komşu düğümlerin karelerinin köşeleriyle bitişiktir. Amacımız sınır düğümlerin kare köşelerinin yabancı komşu düğümlerin kare köşelerine bitiştiği yerleri bulmaktır.

Şekil 4.23 anlatılanı resmetmektedir. Amacımız mavi noktaları bulmaktır. Teknemizde toplam 9 adet düğüm var olduğunu farz edelim. Sol üst taraftaki "baş" kelimesinin yanındaki düğüm ilk sınır düğümümüzdür. Bunun sekiz komşusunu üst taraftaki komşudan başlayarak saat yönüne doğru tek tek bakarız. Eğer baktığımız komşu başak bir damlaya aitse ve çapraz komşulardan biriyse aradaki köşenin pozisyonunu listeye koyarız. Komşulara bakmaya devam ederiz. Baktığımız komşu eğer yanı damlaya ati sınır düğümse yeni baş düğüm bu olarak işaretlenir. Yeni baş düğüm öncekinin çapraz komşusu idiyse aradaki köşe nokta pozisyonu listeye kaydedilir. Yeni baş düğümün komşularını gezmeye önceki baş düğümden itibaren sonrakinin sonrasındaki düğümden başlarız. Süreç bu şekilde ilk düğüme tekrar gelinceye kadar devam eder.

Sınırlar her damla için tespit edildikten sonra bir dizi yumuşatma ve sadeleştirme işleminden sonra SVG [21] biçiminin "path" elemanı şeklinde dosyaya yazdırılır.

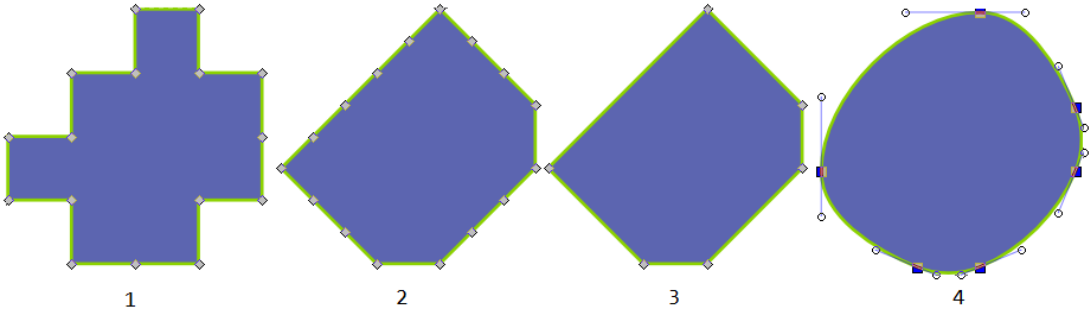


Şekil 4.23 : Sınır düğümlerin ve köşe noktalarının gösterimi. Yeşil noktalar düğümleri, mavi noktalar ise sınırdaki köşeleri gösteriyor.

4.4.1 Kenar Yumuşatma İşlemleri

Kenar yumuşatma işlemleri birbirini takip eden üç adımdan oluşur. Daha fazla yumuşatma gerektiğinde bu adımlar tekrar tekrar kullanılarak yumuşatma artırılabilir. Sonraki paragraflarda adımların ayrıntılı açıklaması verilmiştir.

Yumuşatma adımlarından ilki sınır noktaları doğusallaştırmaya yöneliktir. Izgara ve düğüm yapısının sonucu olarak Şekil 4.18'deki mavi renkle gösterilmiş sınır noktalarının her biri önceki ve sonraki komşu sınır noktalarıyla 90 veya 180 derece açı yapmaktadır. Ardı sıra 90 derece açı yaparak yerleşen noktalar zikzak oluşturmaktadır. Bu zikzaklar göze hoş gelmemektedir. Bu zikzakları basit bit şekilde giderebiliriz. Saat yönünde her bir sınır noktasını komşu sınır noktasına doğru aradaki mesafenin yarısı kadar taşıdığımızda zikzaklar kaybolur. Ayrıca köşelerdeki 90 dereceli açılar da geniş açıya dönüşerek daha akıcı bir şekil oluşur. Şekil 4.24 birinci resim, bir damlanın yumuşatma işlemi öncesi durumunu gösterir. İkinci resim ise her noktanın yarı mesafe taşınması sonrası durumu gösterir.

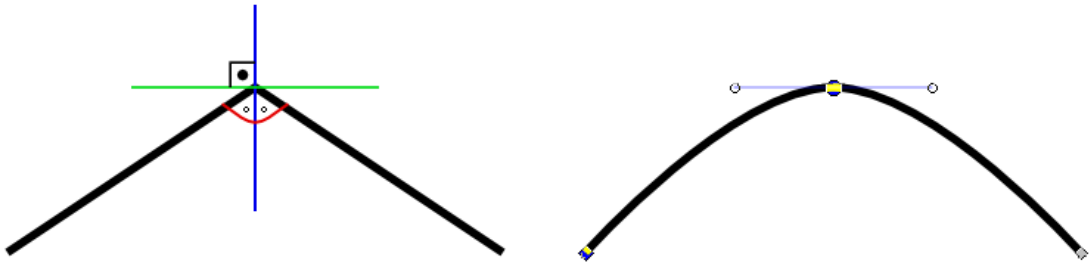


Şekil 4.24 : Kenar yumuşatma işlemleri.

Yumuşatma adımlarından ikincisi ise aynı doğru üzerindeki sınır noktaları aradan kaldırmaktır. Şekil 4.24 ikinci resimde sol üst, sol alt, sağ üst ve sağ alt taraflarda bulunan ve kendi içlerinde aynı doğru üzerinde uzanan noktalar kaldırılarak üçüncü resim elde edilmiştir.

Yumuşatma adımlarından üçüncüsü ise sınır noktalarının Kübik Bezier eğrisinin[11] kontrol noktalarını yeniden düzenleyerek kenarlara eğrisellik kazandırılmasıdır. Eğrinin piksel piksel hangi algoritmalarla çizildiği bu çalışmanın konusu dışında olduğu için ayrıntıya girilmeyecektir. Ana noktalar ve kontrol noktaları uygun bir şekilde belirlense herhangi bir çizim yazılımı bunu çizdirecektir. Kontrol noktalarının pozisyonları belirlenirken iki aşama vardır. Birincisi, noktanın doğrultusunun belirlenmesidir. İkincisi ise bu doğrultuda ne kadar uzağa yerleştirileceğidir.

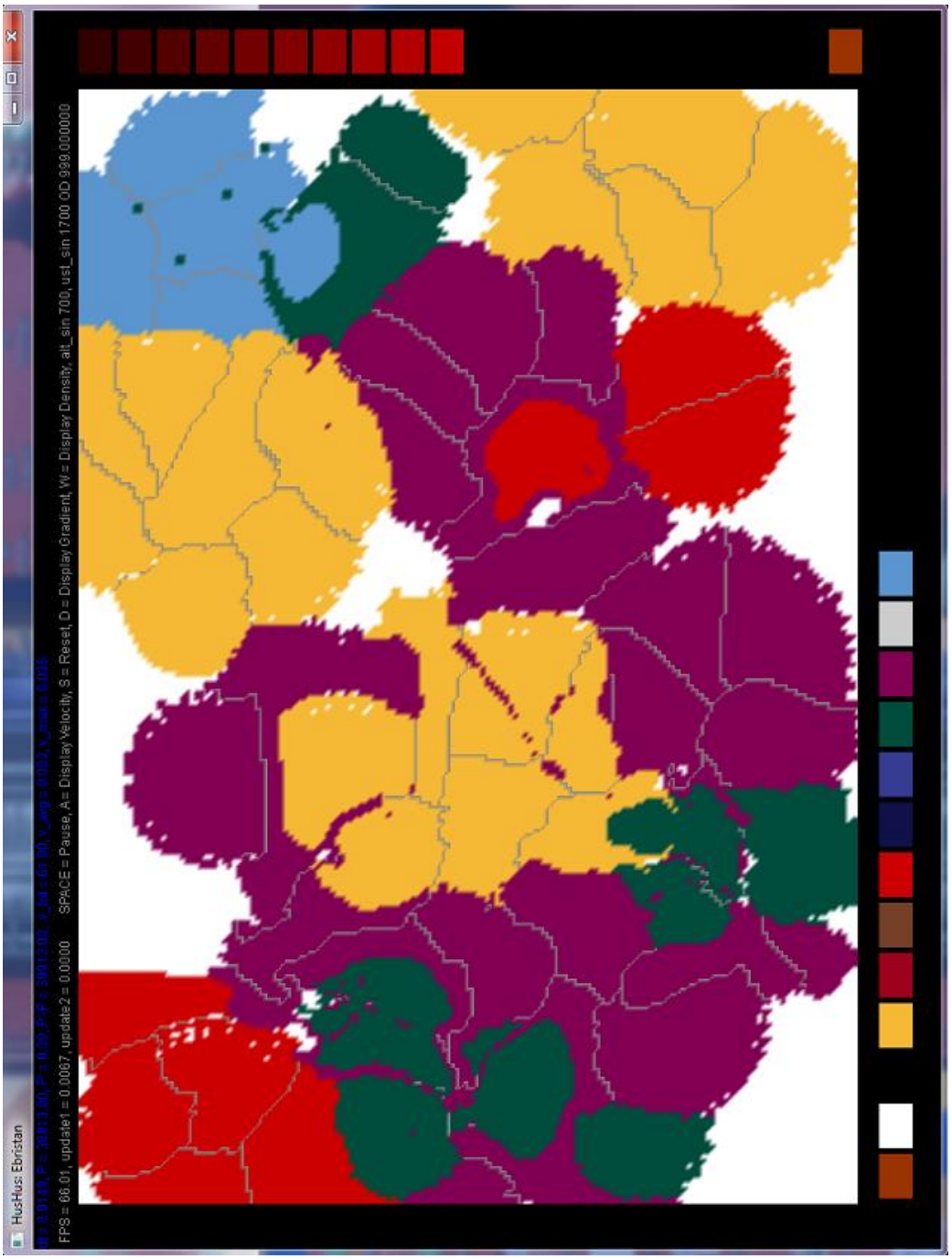
Pürüzsüz bir Spiro eğrisinde[13] yumuşaklık şartlarından birisi bir eğri noktasının her iki kontrol noktasının aynı doğru üzerinde olmasıdır. Bu doğruyu elde ederken ilgili eğri noktasının önceki ve sonraki komşu eğri noktalarıyla yaptığı açıdan yola çıkılır. Bu açıyı ortadan ikiye bölen doğruya dik olan ve ilgili eğri noktasından geçen doğru bizim doğrultumuzu verir. Sağ taraftaki kontrol noktasının bu doğrultu boyunca yerleşeceği uzaklık, ilgili eğri noktasından sağ komşu eğri noktasına olan mesafenin üçte biri kadardır. Sol taraftaki kontrol noktasının aynı doğrultu boyunca yerleşeceği uzaklık, ilgili eğri noktasından sol komşu eğri noktasına olan mesafenin yine üçte biri kadardır. Şekil 4.25 sol taraftaki eğri, doğrultu bulma yöntemini göstermektedir. Ortadaki mavi doğru parçası aradaki açıyı iki eşit parçaya böler. Bu doğru parçasına dik ve ilgili eğri noktasından geçen doğru parçası ise yeşil renkle gösterilmiştir. Sağ taraftaki eğri ise yeşil renkli doğru parçası boyunca aradaki mesafelerin üçte biri oranında uzaklığa yerleştirilen kontrol noktalarını ve yeni oluşan pürüzsüz, yumuşak şekli göstermektedir.



Şekil 4.25 : Yumuşatma işlemlerinin ana şeması.

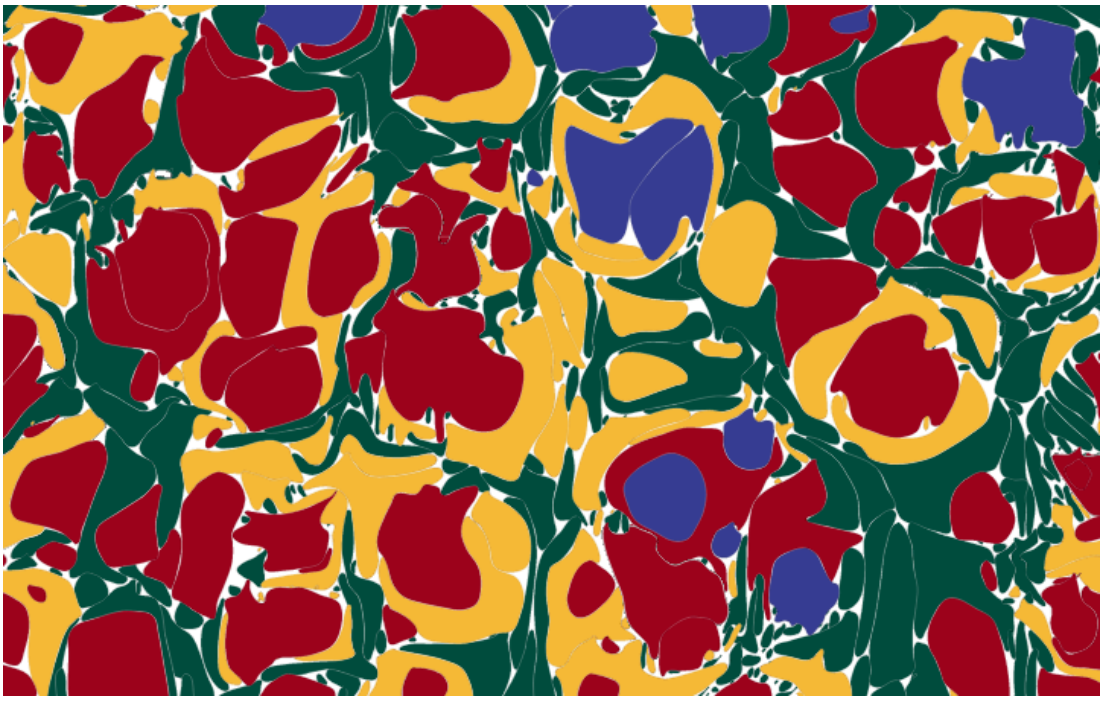
4.5 Yazılımın Tanıtımı ve Kullanımı

Yazılımın arayüzünü Şekil 4.26'da görülmektedir. Ortadaki alana farenin sol tuşu ile tıkladığınızda fırçadan boya serpme eylemini taklit eder. Eğer klavyeden "F" tuşuna bir kez basarsanız biz ile boya eklemeyi taklit eder. Tekrar "F"'ye bastığınıda fırça moduna geçer. Ağşağıdaki farklı renkteki dikdörtgenler boya rengi seççmek için kullanılır. Üzerine gelerek farenin sol tuşuna basarak rengi seçersiniz. "Z" tuşu seçili rengi zemin rengi yapmaya yarar. Sağa ok tuşu vektörel çıktıyı ayrı bir "svg" uzantılı dosya olarak üretir. Sola ok tuşu damlaların yayılışını kütle ve alanları üzerinde oynayarak durdurur. Böylece küçük kıpırdamalar giderilmiş olur. Sağ taraftaki kırmızı dikdörtgenler ise öd değerlerini belirlemeye yarar.

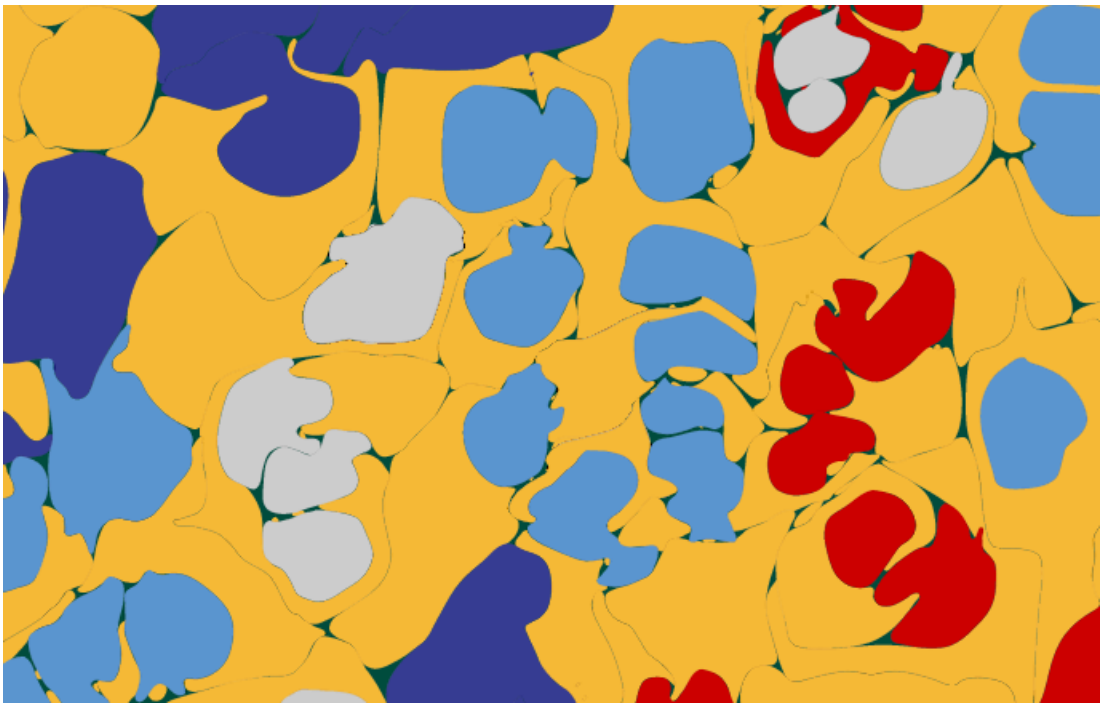


Şekil 4.26 : Yazılımın arayüz görüntüsü

Yazılım kullanılarak elde edilen bazı çıktı örnekleri sırasıyla Şekil 4.27, Şekil 4.28, Şekil 4.29, Şekil 4.30'da verilmiştir.

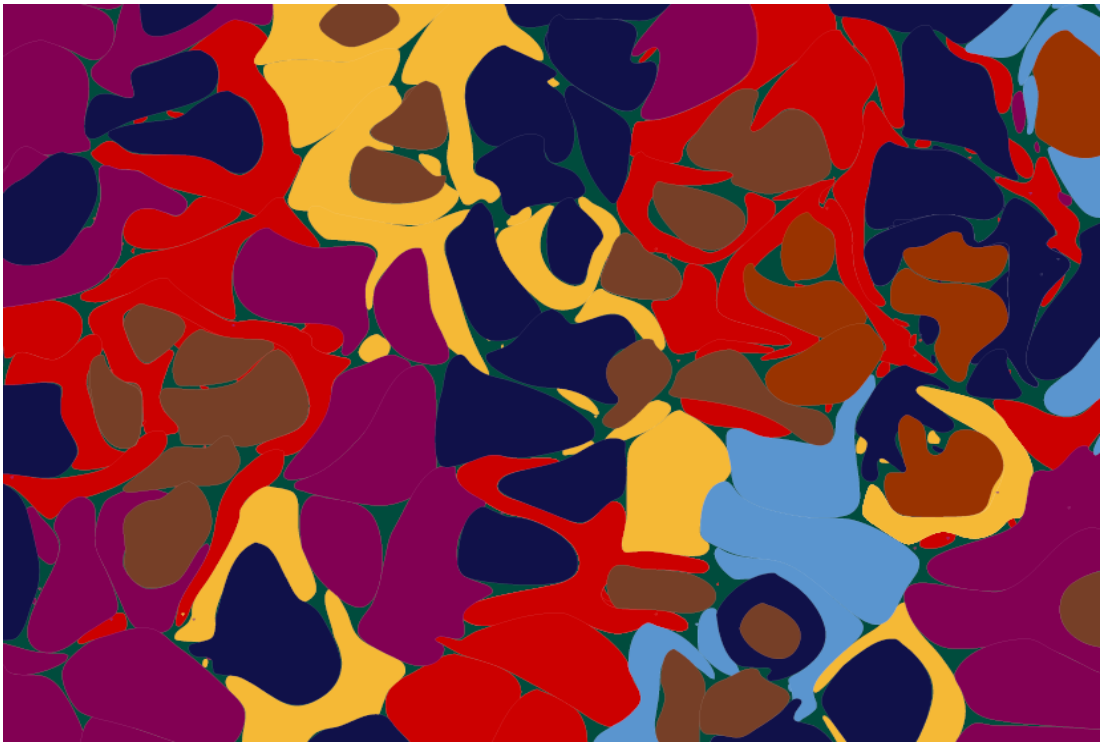


Şekil 4.27 : Örnek çıktılardan birisi.



Şekil 4.28 : Örnek çıktılardan birisi.

Şekil 4.29 : Örnek çıktılardan birisi.



Şekil 4.30 : Örnek çıktılardan birisi.

5. SONUÇ

Tez çalışmamızın sonucunda ortalama bir bilgisayarda sorunsuz çalışabilecek bir yazılım üretilmiştir. Bu yazılım ile Battal Ebrusu çeşitlerine yakın ebru desenleri elde edilebilmektedir. Usta ebrucuların elinden daha güzel desenler çıkacaktır.

5.1 Performans Testleri

Geliştirdiğimiz yazılımın performansını test etmek için farklı donanımlara sahip birkaç bilgisayar kullandık. Çizelge 5.1'deki tabloda performans testi sonuçları verilmiştir. Tablodaki sütunlar, teknenin piksel bazlı boyutuyla düğüm bazlı boyutunun çeşitli kombinasyonlarını içerir. Toplam piksel sayısı piksel bazlı genişlik ve yüksekliğin çarpımıyla bulunur. Düğüm sayısı ise düğüm bazlı genişlik ve yüksekliğin çarpımıyla bulunur. Çözünürlük ise piksel sayısının düğüm sayısına bölümüyle bulunur. Çözünürlük bilgisi, düğüm başına düşen piksel sayısını verir.

Çizelge 5.1'deki sütunlardaki piksel ve düğüm bazlı boyutlar, "K1" başlıklı sütundaki değerlerin, 1.25 veya 1.5 katsayılarıyla çarpılmasıyla elde edilmiştir. Buna göre, K2 sütunundaki piksel bazlı boyut K1 ile aynı olup, düğüm bazlı boyutlar K1'dekilerin 1.25 ile çarpılmasıyla elde edilmiştir. K3 sütunundaki piksel bazlı boyut K1'dekinin 1.25 ile çarpılmasıyla elde edilmiş olup, düğüm bazlı boyut K1'deki ile aynı kalmıştır. K4 sütunundaki tüm boyutlar, K1'dekilerin 1.25 ile çarpımıyla elde edilmiştir. K5 sütunundaki piksel boyutları, K1'dekilerin 1.5 ile çarpılmasıyla elde edilmiş olup, düğüm boyutları K1 ile aynıdır. K6 sütunundaki boyutlar, K1'dekilerin 1.5 ile çarpımıyla elde edilmiştir.

“Çözünürlük” başlıklı satırın altındaki satırlar sırasıyla bilgisayar özellikleri, basınç tabanlı algoritmanın performans değeri ve matematiksel algoritmanın performans değerlerini gösterir. Her bilgisayar için bu üç satır tekrarlanır. Bilgisayar özellikleri satırında, işlemci bilgisi ve grafik kartı bilgisi verilmiştir. Performans değerleri FPS cinsinden verilmiştir. İkili halde bulunan performans değerlerinden sağ taraftaki, tekneye henüz damla damlatılmadan önceki boş halinin performans bilgisini verirken, sol taraftaki ise teknede boş yer kalmayıncaya kadar damlalar serpiştirildiğinde kaydedilen performans bilgisini verir.

Performans test sonuçları bu konuda yapılmış önceki çalışmalarla karşılaştırılmamıştır. Bunun en önemli sebebi, Battal Ebru konusunda yapılmış benzer bir çalışma olmamasıdır. Bize en yakın çalışma olan Lu ve diğ. [10], Battal Ebruyu modellemişlerdir. Hatta yüksek hızlar kaydetmişlerdir. Fakat geliştirdikleri algoritma aslında gerçek zamanlı değildir. Yani kullanıcı boya damlalarının yayılışını veya karıştırılmakta olan teknedeki desenlerin değişimini aynı anda görememektedir. Örneğin, kullanıcı tekneye damla serptiğinde damlalar bir anda yayılır. Burada hız hesabı yapılırken serpme anı ile yayılmanın bittiği an arasındaki süre hesap edilip çarpmaya göre tersi alınarak FPS bulunmalıdır. Çalışmalarında verdikleri FPS değerleri kullanıcı girdisi olmadığı anlarda alınmış gibi görünmektedir. Bu ise algoritmalarının gerçek hızı hakkında şüphelenmemize neden olmuştur.

Çizelge 5.1 : Performans test sonuçları.

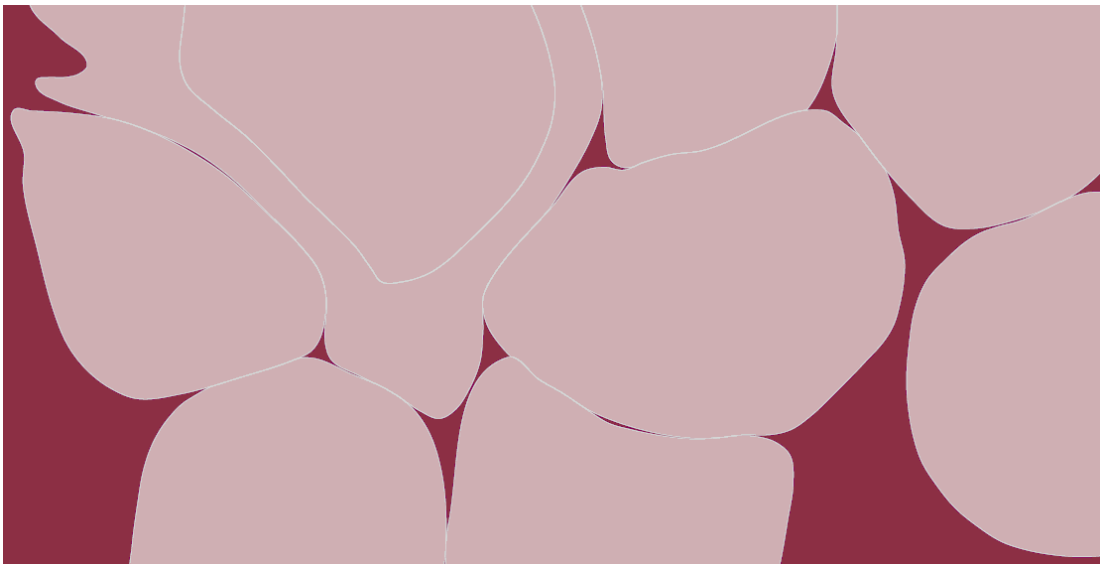
	K1	**K2**	**K3**	**K4**	**K5**	**K6**
Genişlik(piksel)	743	743	928	928	1114	1114
Yükseklik(piksel)	520	520	650	650	780	780
Genişlik(düğüm)	239	298	239	298	239	358
Yükseklik(düğüm)	167	208	167	208	167	250
Çözünürlük(piksel/düğüm)	9,66	6,19	15,1	9,66	21,7	9,66
Bilgisayar 1	Intel(R) Core™2 duo CPU E8500 3.16Ghz(çift çekirdek) – NVIDIA GeForce GTX 560 ekran kartı					
FPS(min-max)	60-80	39-52	60-80	39-52	60-80	27-37
FPS(mat.)(min-max)	74-107	45-75	72-109	42-76	73-110	24-49
Bilgisayar 2	Intel(R) Core™2 duo CPU E8500 3.16Ghz(çift çekirdek) – ekran kartı yok					
FPS(min-max)	54-72	34-47	54-70	35-47	55-68	25-33
FPS(mat.) (min-max)	72-99	50-64	72-99	49-63	72-98	28-44
Bilgisayar 3	Intel(R) Core™2 duo CPU E8500 3.16Ghz(çift çekirdek) – Ati Radeon HD 4600 Series					
FPS(min-max)	54-73	35-48	54-72	35-49	56-73	25-33
FPS(mat.) (min-max)	54-68	35-56	57-68	35-56	58-66	28-38
Bilgisayar 4	Intel(R) Core™ i3-2100 CPU 3.10Ghz(dört çekirdek) - Ati Radeon HD					
FPS(min-max)	69-115	49-72	74-115	47-72	73-115	37-49
FPS(mat.) (min-max)	91-166	65-108	90-166	68-107	87-167	37-75
Bilgisayar 5	Intel core i7-2630QM CPU 2.0 GHz-nvidia geforce 540m ekran karti					
FPS(min-max)	76-135	50-85	80-134	51-85	75-138	34-51
FPS(mat.) (min-max)	118-246	88-157	100-245	87-158	128-240	47-84

Performans test sonuçlarına bakarak bazı genel yargılara ulaşılmıştır.

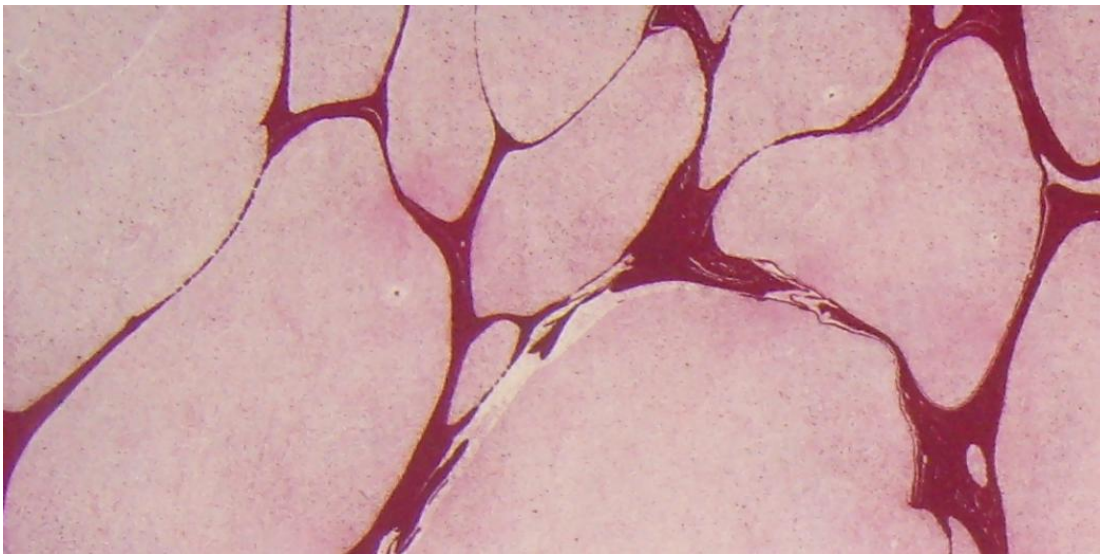
- Her iki algoritma için düğüm sayısının artmasıyla birlikte performansta düşme olmaktadır. Bu düşme ters orantılıdır. Örneğin, K1 ve K2'deki tekne piksel sayıları aynı fakat düğüm sayıları farklıdır. Bu iki ayarın performans oranları 60/39 = 1.54 olurken düğüm sayıları oranı ise 39913/61984 = 0.64 olmaktadır. 0.64'ün çarpa işlemine göre tersini aldığımızda, yani düğüm sayısı büyük olanı küçük olana böldüğümüzde 1.55 sayısını buluruz. Buradan vardığımız yargı: düğüm sayıları aynı olduğu sürece tekne boyutundaki artışların performansa etkisi ihmal edilecek kadar azdır. Örneğin, K1, K2 ve K5 sütunlarının düğüm sayıları aynı fakat piksel sayıları farklıdır. Buna rağmen performansları aynıdır.
- Bir teknenin piksel boyutu aynı iken düğüm sayısı artırıldıkça damlaların görsel kalitesi ve tekneye atılabilecek damla sayısı da artar. Görsel kalitenin ölçüsünü çözünürlük verir. Tablodaki çözünürlük değeri küçüldükçe teknedeki ayrıntı artar.
- Matematiksel algoritmanın performansı tüm bilgisayarlar için basınç tabanlı algoritmadan daha iyidir. Bilgisayar 1, Bilgisayar 2 ve Bilgisayar 3'ün işlemci tipleri aynı, ekran kartları farklıdır. Bilgisayar 1 çok daha iyi bir ekran kartına sahiptir. Fakat bu üç bilgisayarın performansları arasında dikkate değer bir fark yoktur. Böyle olması anlamlıdır. Çünkü algoritmalar, GPU'yu direk kullanmamaktadır. Yalnızca DirectX [19] kütüphanesi, ekrana çizdirme yaparken GPU'yu kullanmaktadır. Ekrana çizdirme işlemleri masraflı olmadığı için grafik kartının etkisi ihmal edilecek kadar azdır.

- Bilgisayarlar arasında performansa etki eden en önemli etkenlerden birisi işlemcilerindeki çekirdek sayısıdır. Bilgisayar 1,2 ve 3 çift çekirdekli, bilgisayar 4, dört çekirdekli, bilgisayar 5 ise sekiz çekirdeklidir. Bu bilgisayarların çekirdek sayılarıyla doğru orantılı olarak performansları artmaktadır. Algoritmalar paralel programlama ile geliştirilmemesine rağmen oluşan bu doğru orantı bir açıklamaya ihtiyaç duymaktadır. Bilgisayarda işlemci, birçok program tarafından paylaşılmaktadır. İşletim sistemi bu paylaştırma işi için çeşitli yöntemler kullanmaktadır. Birden çok çekirdeğe sahip işlemcilerde program başına düşen işlemci sayısı artacağı için aynı anda çalışmakta olan programların performansı artacaktır. Bu artış düzgün bir orantı ile değildir. Çünkü işletim sistemi, hangi programın nasıl bir algoritması olduğunu bilemeyeceği için daha genel bir paylaşım yapar. Eğer bir algoritma paralel programlama ile geliştirildiyse işlemcilerden maksimum performansı alır.
- Performans açısından genel kabul görmüş olan alt limit 25 FPS'dir. Düğüm sayısı 358x250 = 89500 olduğunda FPS değeri 25'e kadar düştüğü görülmüştür. Bu düğüm sayısı alt limitimizdir. Teknenin piksel sayısının performansa etkisi ihmal edilebilecek kadar az olduğu göz önünde bulundurulursa, daha büyük piksel bazlı boyutlar arzu edildiğinde tekne piksel genişlik ve yükseklik değerleri rahatlıkla artırılabilir.

Sonuç değerlendirmesinde bahsedilmesi gereken önemli bir konu da üretilen ebru çıktılarının görsel kalitesidir. Tekne içindeki damla öbeklerinin sınırlarının güzel çizdirilebilmesi çalışmanın başarısı açısından olmazsa olmaz bir özelliktir. Çalışma sonucunda geliştirdiğimiz yazılımla oluşturulmuş bir çıktıdan alınan kesit Şekil 5.1'de verilmiştir. Bu şekille aynı renklerin kullanıldığı gerçek bir ebru çıktısı ise Şekil 5.2'de verilmiştir. İki şekil arasında, damlaların birbirini itmesi a ve damla sınırlarının düzgün eğriler şeklinde olması açılarından çok benzer olmasına dikkat çekeriz. Kullanılan yumuşatma yöntemi [13] gayet iyi bir iş başarmıştır.



Şekil 5.1 : Tez çalışmasında geliştirilen yazılımla üretilen bir ebrudan kesit.



Şekil 5.2 : Gerçek bir ebru çıktısından alınmış kesit.

5.2 Tavsiyeler ve Gelecek Çalışmalar

Çalışmamız sonucunda şimdilik sadece Battal Ebrusu deseni yapılabilmektedir. Bunda birçok iyileştirmeye ihtiyaç vardır. Örneğin, Battal Ebru desenlerinin biraz daha eliptik şekil alması gerekmektedir. Sonra, biz aleti yardımıyla karıştırma özelliğinin eklenmesi olmazsa olmazdır. Algoritmanın daha hızlı olması için optimize edilmesi gerekmektedir. Mobil cihazlarda da iyi performans vermesi için yeniden düzenlenmelidir. Vektörel çıktının bir tuşa basılarak aynı tekne içinde görülmesi sağlanmalıdır. Yazılım ara yüzüne bazı yardımcı araçlar eklenebilir.

Algoritmanın, GPU ve CPU üzerinde paralel programlama tekniğiyle yeniden uyarlanması mevcut işleme gücünden azami fayda sağlamak açısından önemli bir adım olacaktır. Ayrıca, çalışmamızda geliştirilen yöntemleri bir yayına dönüştürmek için çalışmalar yapılacaktır.

KAYNAKLAR

[1] **Euler Denklemleri**, (t.y.) Wikipedia'dan, Alındığı tarih: 15.05.2011, Adres: http://en.wikipedia.org/wiki/Euler_equations_(fluid_dynamics)

[2] **Navier,C. L. M. H.,** 1822: "Memoire sur les lois du mouvement des fluides", *Mem. Acad. Sci. Inst. France, 6, 389-440*.

[3] **Stam,J.,** 1999: Stable fluids. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, p.121-128

[4] **Stam,J.**, 2003: Real-Time Fluid Dynamics for Games. *Proceedings of the Game Developer Conference*

[5] **ORUÇ, Ç., KARABAY, I., ÖREN, D.**, 2003: A Physical Study of The Art of Marbling, *Journal of Yildiz Technical University*, p.19-27.

[6] **ACAR, R., AND BOULANGER, P.**, 2006: Digital marbling: A multiscale fluid model. *IEEE Transactions on Visualization and Computer Graphics* 12, 4, 600–614.

[7] **ACAR, R.,** 2005: digital Marbling Based on Computational Fluid Dynamics, *Department of Computing Science, University of Alberta* (Doktora Tezi), Alındığı yer: yazarın kendisinden

[8] **Jin, X., Chen, S., Mao, X.**, 2007: Computer-generated marbling textures: a GPU-based design system. *IEEE Comput Graph Appl* 27(2):78–84

[9] **Ando, R., and Tsuruno, R.,** 2010: Vector Fluid: A Vector Graphics Depiction of Surface Flow, *Proceeding NPAR '10 Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, p.129-135

[10] **Lu, S., Jaffer, A., Jin, X., Zhao, H., and Mao, X.,** 2011: Mathematical Marbling, *IEEE Computer Graphics and Applications, vol. 99, no. PrePrints*,

[11] **Farin, G.,** 1997: Curves and surfaces for computer-aided geometric design (4 ed.), *Elsevier Science & Technology Books, ISBN 978-0-12-249054-5*

[12] **Url-1** <*www.inkscape.org*>, Alındığı tarih: 10.04.2012

[13] **Levien, R. L.,** 2009: From Spiral to Spline: Optimal Techniques in Interactive Curve Design By Raphael linus Levien, *University of California, Berkeley*Technical Report No. UCB/EECS-2009-162 (Doktora Tezi), alındığı adres: *<http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-162.html>*

[14] **Derman, M. U.,** 1977: "Türk Sanatında Ebru" , *Akbank Press, İstanbul*.

[15] **Derman, M.U.,** (t.y.) alındığı tarih: 07.08.2011, Osmanlı Ansiklopedisi. C.11,s.189

[16] **Url-2, Dinçer, K.** (t.y.), Alındığı tarih: 04.04.2012, Adres: http://www.ebrusitesi.com/index.htm

[17] **Bridson, R.,** 2008: Fluid Simulation for Computer Graphics, *A K Peters/CRC Press* - 246 Sayfa

[18] **Butts, A.** 2005: Lecture 16: Fluid Simulation in Computer Graphics, *Department of Computer Science, Cornell University*,CS667 Ders Notu, Alındığı tarih: 15.04.2011, Adres: http://www.cs.cornell.edu/courses/cs667/2005sp/notes/16butts.pdf

[19] **Url-3, DirectX,** <http://www.microsoft.com/en-us/download/details.aspx?id=35>, *Microsoft Corporation*, Alındığı tarih: 24.02.2011

[20] **Url-4 West, M.,** <*http://cowboyprogramming.com/2008/04/01/practical-fluid-mechanics/*>, Alındığı tarih: 02.03.2011

[21] **Url-5** <*http://www.w3.org/TR/SVG/paths.html*>, Alındığı tarih: 14.04.2012

[22] **Url-6 OpenGL**,<http://www.opengl.org/>, Alındığı tarih: 01.04.2011

[23] **Url-7 Bilineer İnterpolasyon**, (t.y.) Wikipedia'dan, Alındığı tarih: 05.03.2012, Adres: http://en.wikipedia.org/wiki/Bilinear_interpolation

[24] **Xu,J., Mao,X., and Jin,X.** 2008: "Nondissipative marbling," *IEEE Computer Graphics and Applications*, vol. 28, no. 2, pp. 35–43.

[25] **Zhao,H., Jin,X., Lu,S., Mao,X., and Shen,J.,** 2009: "AtelierM++: a fast and accurate marbling system," *Multimedia Tools and Applications*, vol. 44, no. 2, pp. 187–203.

[26] **Ando, R., and Tsuruno, R.**, 2011: Vector graphics depicting marbling flow, *Computers & Graphics* , Vol. 35 no. 1, pp. 148-159.

ÖZGEÇMİŞ

Ad Soyad: Hüseyin SAVRAN

Doğum Yeri ve Tarihi: DENİZLİ – 11.04.1984

Adres: Yalova Üniversitesi Bilgisayar Mühendisliği Bölümü

Lisans Üniversite: Yeditepe Üniversitesi

Yayın Listesi:

- **Capacity Leasing and Operation Allocation Issues in Telecommunication Networks under Fuzzy Quality of Service Constraints (TSYS-2011-0475)** Hasan Turan, Nihat Kasap, **Hüseyin Savran**, International Journal of Systems Science
- Hasan Huseyin Turan, **Huseyin Savran** and Nihat Kasap,"**Provider Selection and Task Allocation Model for Telecommunication Networks under Quality of Service Degradation Policies**",31st National Conference on Operational Research and Industrial Engineering, July 5-7, 2011, Sakarya, Turkey.
- Hasan Huseyin Turan, Nihat Kasap and **Huseyin Savran**, "**Provider Selection and Task Allocation Problems under Fuzzy Quality of Service Constraints and Volume Discount Pricing Policy for Telecommunication Network**", 2st International Symposium on Computing in Science & Engineering (ISCSE 2011), June 1-4, 2011, Izmir, Turkey.
- **Savran, H.** and Panfilov, **P.B. 3D Head Model Reconstruction from Two Orthogonal Views**.Proc. Int'l Symposium on New Information Technologies and Quality Management (NIT&MQ 2008). Belek, Turkey, 16-23 May 2008, pp.155-158. (in Russian).

Projeler:

- **Araştırmacı**, **Saysal ebru yazılımı geliştirilmesi** , Istanbul Büyükşehir Belediyesi, PROJEM İSTANBUL, 2011-2012
- **Araştırmacı**, **Sourcing Strategies Under Quality of Service Requirements and Different Pricing Schemes for Telecommunication Networks**, Junior Researcher Career Development Program, The Scientific and Technological Research Council of Turkey (TÜBITAK), Grant No: 106K263, 2009-2011
- **Araştırmacı**, **Provider Selection and Task Allocation Problems under Fuzzy Quality of Service Constraints and Volume Discount Pricing Policy for Telecommunication Network**, Short Term R&D Funding Programme ,The Scientific and Technological Research Council of Turkey (TÜBITAK), Grant No: 110K609, 2011-2012