BURSA TEKNIK ÜNIVERSITESI

BURSA TEKNİK ÜNİVERSİTESİ MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

❖ **DERS**: Algoritma ve Programlama ❖

❖ **PROJE KONUSU**: CLI - Uzay Simülasyonu ❖

❖ **ADI SOYADI**: Deniz BAŞAT | Şube 1 ❖

❖ **NUMARA**: 24360859921 ❖

❖ **BÖLÜM**: Bilgisayar Mühendisliği ❖

❖ **GITHUB**: TIK* ❖

1. GİRİŞ

Mühendislik eğitiminde, teorik bilginin pratik uygulamaya dökülmesi, algoritmik düşünme yetisinin kazanılması açısından hayati önem taşımaktadır. Bu proje raporu, **Bursa Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü** müfredatında yer alan "Algoritmalar ve Programlama" dersi kapsamında, 2025-2026 Güz Dönemi final projesi olarak hazırlanan **"Uzay Simülasyonu"** uygulamasını tanıtmaktadır[1111].

Projenin temel amacı; C programlama dilinin güçlü ve düşük seviyeli özelliklerini kullanarak, Güneş Sistemi'mizde yer alan 8 farklı gezegenin (Merkür, Venüs, Dünya, Mars, Jüpiter, Satürn, Uranüs ve Neptün) çekim alanlarındaki fiziksel değişimleri simüle etmektir[22222222]. Geliştirilen yazılım, kullanıcıyı bir "Bilim İnsanı" olarak konumlandırmakta ve çeşitli parametreler (kütle, hız, zaman vb.) girilerek evrensel fizik yasalarının farklı yerçekimi ivmeleri (g) altında nasıl sonuçlar ürettiğini analiz etmesine olanak tanımaktadır[3333].

Bu proje sadece fiziksel formüllerin (Serbest Düşme, Arşimet Prensibi, Hidrostatik Basınç vb.) kodlanmasından ibaret değildir[4444]. Yazılımın arka planında, bilgisayar bilimlerinin temel taşlarından olan **bellek yönetimi (memory management)** ve **işaretçi aritmetiği (pointer arithmetic)** konularının derinlemesine uygulanması hedeflenmiştir[5]. Proje teknik şartnamesi gereği, veri yapılarına erişimde geleneksel dizi indeksleme yöntemleri ($dizi[i]$) yerine, doğrudan bellek adresleri üzerinden işlem yapan işaretçi mantığı ($*(ptr+i)$) kullanılmıştır[6]. Bu kısıtlama, donanıma yakın programlama becerilerini geliştirmeyi amaçlamaktadır.

Uygulama, herhangi bir grafiksel arayüz (GUI) barındırmayan, tamamen metin tabanlı (konsol) bir yapıya sahiptir[7]. Ancak modüler yapısı, kullanıcı dostu menü tasarımı ve hatalı veri girişlerine karşı geliştirilen koruma mekanizmaları (Ternary Operator kullanımı gibi) sayesinde profesyonel bir yazılım mimarisine sahiptir[8].

Bu proje, proje dokümanında belirtilen kurallar ve akademik dürüstlük ilkeleri çerçevesinde, hiçbir dış kaynak kod kullanılmadan ve başka bir projeden kopyalanmadan tamamen bireysel olarak geliştirilmiştir[9]

2. TEKNİK DETAYLAR

Bu bölümde, projenin yazılım mimarisi, bellek yönetim stratejileri, veri yapıları ve uygulanan algoritmaların teknik ayrıntıları ele alınmıştır. Proje geliştirme sürecinde, kodun okunabilirliğini ve bakımını kolaylaştıran "Modüler Programlama" ve "Temiz Kod" (Clean Code) prensiplerine sadık kalınmıştır.

2.1. Program Akışı ve Modüler Yapı

Yazılım, tek bir kaynak dosyası yerine, işlevsel kümelerine göre ayrılmış dosyalardan oluşan modüler bir mimariye sahiptir. Bu yaklaşım, kod karmaşıklığını azaltmış ve hata ayıklama (debugging) süreçlerini hızlandırmıştır. Projenin dosya yapısı ve işlevleri aşağıda detaylandırılmıştır:

- main.c (Ana Modül): Programın giriş noktasıdır (Entry Point). Temel görevi, programın yaşam döngüsünü yönetmektir. Kullanıcıdan "Bilim İnsanı" ismini alır, gerekli bellek tanımlamalarını yapar ve ana menü döngüsünü (while loop) başlatır. Kullanıcı arayüzü (UI) etkileşimleri burada yönetilir.
- config.h (Konfigürasyon Dosyası): Projenin veri tabanı katmanını oluşturur. Güneş Sistemi'ndeki gezegenlerin isimleri (string array) ve bu gezegenlere ait yerçekimi ivmesi değerleri (double array) burada sabit (static) diziler olarak tanımlanmıştır. Bu yapı sayesinde, yerçekimi verilerinin ana kod içerisinde "sihirli sayılar" (magic numbers) olarak dağınık durması engellenmiştir.
- utils.h (Araçlar ve Hesaplama Kütüphanesi): Uygulamanın iş mantığını (Business Logic) barındırır. Tüm fiziksel formüllerin kodlandığı fonksiyonlar, metin işleme yardımcıları ve ekran biçimlendirme araçları bu başlık dosyasında toplanmıştır. main.c dosyası bu kütüphaneyi dahil ederek (#include) fonksiyonlara erişir.

Program Çalışma Algoritması:

1. Başlatma: Program derlenip çalıştırıldığında, öncelikle gezegen verileri belleğe yüklenir.
2. Kimlik Doğrulama: Kullanıcıdan isim istenir ve bu veri fgets fonksiyonu ile güvenli bir şekilde alınır.
3. Sonsuz Döngü: Program, kullanıcı çıkış komutunu (-1) verene kadar do-while veya while(1) döngüsü içerisinde kalır.
4. Seçim ve Yürütme: Her döngü adımında menü ekrana basılır. Kullanıcının seçimi switch-case yapısı ile analiz edilir ve ilgili deney fonksiyonu çağrılır.
5. Sonlandırma: Çıkış komutu ile döngü kırılır ve program return 0 ile sonlanır.

2.2. Pointer (İşaretçi) ve Dizi Kullanımı

Proje teknik şartnamesinin en kritik maddesi olan "Dizi indekslemesinin yasaklanması" kuralı, projenin bellek yönetimi stratejisini belirlemiştir. C dilinde diziler, bellekte ardışık bloklar halinde tutulur ve dizi adı aslında ilk elemanın adresini gösteren bir işaretçidir (pointer).

Bu projede, dizi[i] şeklindeki geleneksel erişim yöntemi tamamen devre dışı bırakılmış, bunun yerine Pointer Aritmetiği (Pointer Arithmetic) kullanılmıştır.

Uygulanan Teknik: Verilere erişim, temel adres (base address) üzerine ofset (kaydırma) değeri eklenerek sağlanmıştır.

- Formül: Adres = Başlangıç Adresi + (İndeks * Veri Tipi Boyutu)
- Kod İmplementasyonu: double yercekim_ivmesi = *(g_ptr + i); char *gezegen_adi = *(isim_ptr + i);

Bu yöntem sayesinde:

1. Öğrencinin bellek adresleme mantığına hakimiyeti pekiştirilmiştir.
2. Fonksiyonlara büyük dizilerin kopyalanması yerine, sadece başlangıç adresleri (8 byte) gönderilerek Call-by-Reference yöntemi uygulanmış ve bellek performansı artırılmıştır.

2.3. Girdi Doğrulama ve Hata Yönetimi

Fiziksel simülasyonların doğruluğu, girilen verilerin tutarlılığına bağlıdır. Kütle (m), zaman (t), uzunluk (L) ve hacim (V) gibi skaler fiziksel büyüklüklerin negatif olması fizik kurallarına aykırıdır.

Kullanıcı deneyimini (UX) iyileştirmek ve programın çökmesini (crash) engellemek adına, "Girdi Sanitizasyonu" (Input Sanitization) uygulanmıştır. Proje isterlerine uygun olarak, bu kontrol için dallanma yaratan if-else blokları yerine, daha kompakt ve hızlı çalışan Ternary Operator (Üçlü Operatör) tercih edilmiştir.

Kod Örneği ve Mantığı: t = (t < 0) ? -t : t;

Bu satır, işlemci seviyesinde koşullu atama yaparak, kullanıcının hatasını (örn: -10 saniye girişi) otomatik olarak düzeltir (10 saniye) ve simülasyonun kesintisiz devam etmesini sağlar.

2.4. Deneylerin Hesaplama Mantığı

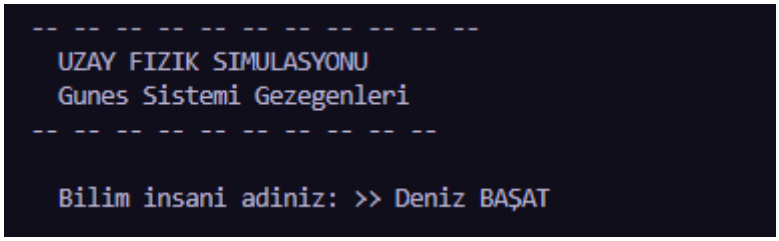
Simülasyon motoru, Newton Mekaniği ve Akışkanlar Dinamiği prensiplerine dayanan 9 farklı deneyi kapsamaktadır. Her deney bir fonksiyon olarak tanımlanmış olup, parametre olarak aldığı gezegen verileri üzerinde iterasyon yaparak (0'dan 7'ye kadar) sonuç üretir. Kullanılan formüller ve açıklamaları şöyledir:

1. Serbest Düşme Deneyi: Havasız ortamda serbest bırakılan cismin aldığı yol hesaplanır.
    - Formül: $h = (1/2) * g * t^2$
    - Birim: Metre (m)
2. Yukarı Atış Deneyi: Belirli bir ilk hızla (v0) dikey fırlatılan cismin çıkabileceği maksimum irtifa bulunur.
    - Formül: $h\_max = (v0^2) / (2 * g)$
    - Birim: Metre (m)
3. Ağırlık Deneyi: Kütlenin (m), farklı çekim ivmelerindeki kuvvet karşılığı hesaplanır.
    - Formül: G = m * g
    - Birim: Newton (N)
4. Kütleçekimsel Potansiyel Enerji: Cismin konumundan kaynaklı depoladığı enerji miktarıdır.
    - Formül: Ep = m * g * h
    - Birim: Joule (J)
5. Hidrostatik Basınç: Sıvıların, derinliğe (h) ve yoğunluğa (rho) bağlı olarak uyguladığı taban basıncıdır.
    - Formül: P = rho * g * h
    - Birim: Pascal (Pa)
6. Arşimet Kaldırma Kuvveti: Sıvı içerisine batan bir cisme etki eden yukarı yönlü kuvvettir.
    - Formül: Fk = rho * g * V
    - Birim: Newton (N)
7. Basit Sarkaç Periyodu: İpe bağlı bir kütlenin tam salınım süresidir. Hesaplamada karekök işlemi için math.h kütüphanesinden sqrt() fonksiyonu kullanılmıştır.
    - Formül: T = 2 * PI * sqrt(L / g)
    - Birim: Saniye (s)

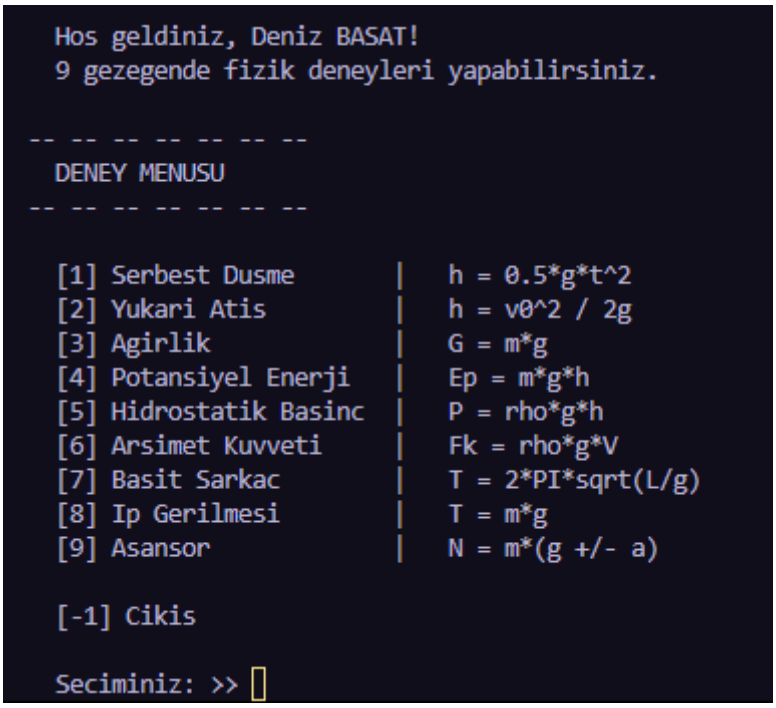
8. Sabit İp Gerilmesi: Durgun halde asılı duran bir kütlenin ipte oluşturduğu gerilme kuvvetidir.
    - Formül: T = m * g
    - Birim: Newton (N)
9. Asansör Deneyi (Eylemsizlik): İvmeli hareket eden referans sistemlerinde cismin "Görünür Ağırlığının" değişimidir.
    - Yukarı Hızlanma / Aşağı Yavaşlama: N = m * (g + a)
    - Aşağı Hızlanma / Yukarı Yavaşlama: N = m * (g - a)
    - Birim: Newton (N)

3. EKRAN ÇIKTILARI

Aşağıda programın çalışma anına ait ekran görüntüleri yer almaktadır.



Şekil 1. *Program Açılışı ve İsim İst*



Şekil 2. *Ana Menü Görünümü*

```
Seciminiz: >> 1

Zaman (t) giriniz [saniye]: 3600
Girilen deger: 3600.00 s

[ SERBEST DUSME ]
------------------------------------------------------------
| Gezegen      |                        Yukseklik (m) |
------------------------------------------------------------
| Merkur       |                  23976000.0000 m     |
| Venus        |                  57477600.0000 m     |
| Dunya        |                  63568800.0000 m     |
| Mars         |                  24040800.0000 m     |
| Jupiter      |                 160639200.0000 m     |
| Saturn       |                  67651200.0000 m     |
| Uranus       |                  56311200.0000 m     |
| Neptun       |                  72252000.0000 m     |
------------------------------------------------------------
```

Şekil 3. Serbest Düşme Deneyi Sonuçları

```
Seciminiz: >> 3

Kutle (m) giriniz [kg]: 80.1
Yukseklik (h) giriniz [m]: 1.75
Girilen kutle: 80.10 kg, yukseklik: 1.75 m

[ POTANSIYEL ENERJI ]
------------------------------------------------------------
| Gezegen      |                          Enerji (J) |
------------------------------------------------------------
| Merkur       |                    518.6475 J     |
| Venus        |                   1243.3522 J     |
| Dunya        |                   1375.1167 J     |
| Mars         |                    520.0493 J     |
| Jupiter      |                   3474.9382 J     |
| Saturn       |                   1463.4270 J     |
| Uranus       |                   1218.1208 J     |
| Neptun       |                   1562.9513 J     |
------------------------------------------------------------
```

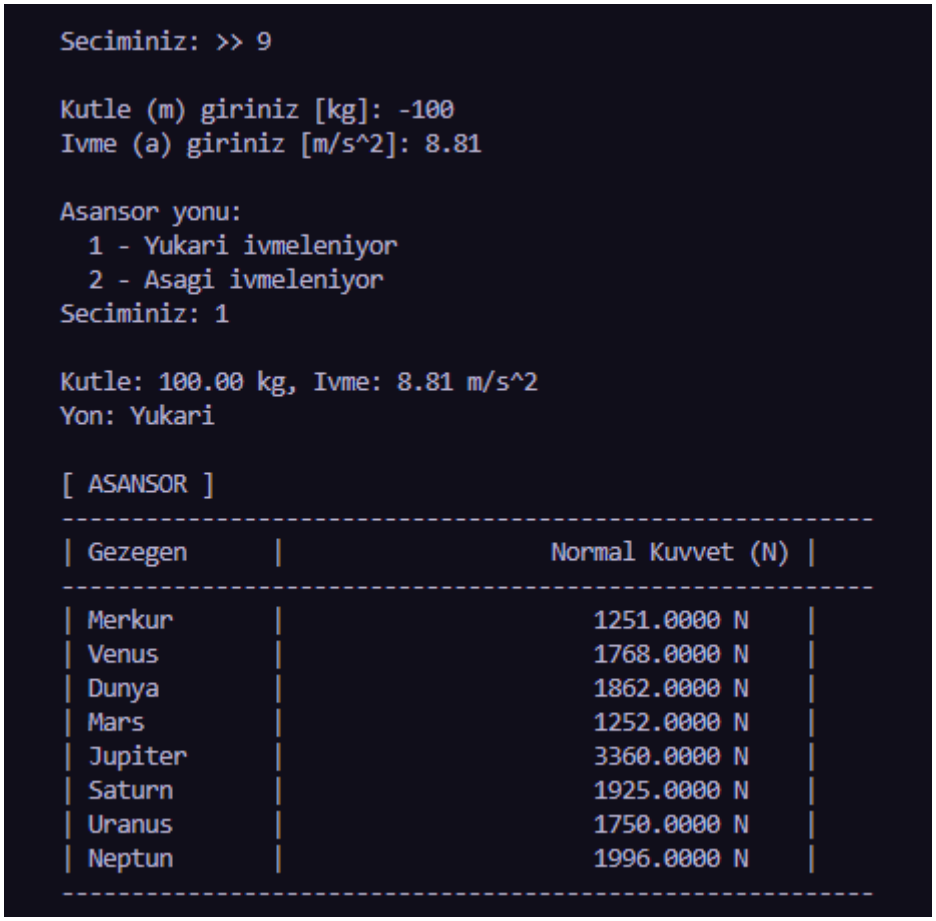
Şekil 4. Ağırlık Deneyi Sonuçları

```
Seciminiz: >> 7

Sarkac uzunlugu (L) giriniz [m]: 5
Girilen deger: 5.00 m

[ BASIT SARKAC ]
-----------------------------------------------------------
| Gezegen      |                          Periyot (s) |
-----------------------------------------------------------
| Merkur       |                          7.3041 s    |
| Venus        |                          4.7174 s    |
| Dunya        |                          4.4857 s    |
| Mars         |                          7.2942 s    |
| Jupiter      |                          2.8218 s    |
| Saturn       |                          4.3483 s    |
| Uranus       |                          4.7660 s    |
| Neptun       |                          4.2075 s    |
-----------------------------------------------------------
```

Şekil 5. Basit Sarkaç Deneyi Sonuçları

```
Seciminiz: >> 9

Kutle (m) giriniz [kg]: 89
Ivme (a) giriniz [m/s^2]: 8.81

Asansor yonu:
  1 - Yukari ivmeleniyor
  2 - Asagi ivmeleniyor
Seciminiz: 2

Kutle: 89.00 kg, Ivme: 8.81 m/s^2
Yon: Asagi

[ ASANSOR ]
-----------------------------------------------------------
| Gezegen      |                   Normal Kuvvet (N) |
-----------------------------------------------------------
| Merkur       |                      -454.7900 N    |
| Venus        |                         5.3400 N    |
| Dunya        |                        89.0000 N    |
| Mars         |                      -453.9000 N    |
| Jupiter      |                      1422.2200 N    |
| Saturn       |                       145.0700 N    |
| Uranus       |                       -10.6800 N    |
| Neptun       |                       208.2600 N    |
-----------------------------------------------------------
```

Şekil 6. Asansör Deneyi ve Yön Seçimi



```
Seciminiz: >> 9

Kutle (m) giriniz [kg]: -100
Ivme (a) giriniz [m/s^2]: 8.81

Asansor yonu:
  1 - Yukari ivmeleniyor
  2 - Asagi ivmeleniyor
Seciminiz: 1

Kutle: 100.00 kg, Ivme: 8.81 m/s^2
Yon: Yukari

[ ASANSOR ]
-----------------------------------------------------
| Gezegen      |                  Normal Kuvvet (N) |
-----------------------------------------------------
| Merkur       |                     1251.0000 N    |
| Venus        |                     1768.0000 N    |
| Dunya        |                     1862.0000 N    |
| Mars         |                     1252.0000 N    |
| Jupiter      |                     3360.0000 N    |
| Saturn       |                     1925.0000 N    |
| Uranus       |                     1750.0000 N    |
| Neptun       |                     1996.0000 N    |
-----------------------------------------------------
```

Şekil 7. Negatif Değer Kontrolü (Ternary Operator)

```
-- -- -- -- -- -- --
  DENEY MENUSU
-- -- -- -- -- -- --

  [1] Serbest Dusme       |   h = 0.5*g*t^2
  [2] Yukari Atis         |   h = v0^2 / 2g
  [3] Agirlik             |   G = m*g
  [4] Potansiyel Enerji   |   Ep = m*g*h
  [5] Hidrostatik Basinc  |   P = rho*g*h
  [6] Arsimet Kuvveti     |   Fk = rho*g*V
  [7] Basit Sarkac        |   T = 2*PI*sqrt(L/g)
  [8] Ip Gerilmesi        |   T = m*g
  [9] Asansor             |   N = m*(g +/- a)

  [-1] Cikis

  Seciminiz: >> -1

  Gule gule, Deniz BASAT!
```

Şekil 8. Programdan Çıkış

4. EKSİKLİKLER VE GELİŞTİRMELER

Geliştirilen "Uzay Fizik Simülasyonu" projesi, mevcut haliyle proje dokümanında belirtilen tüm temel isterleri (modüler yapı, pointer aritmetiği kullanımı, hata yönetimi vb.) eksiksiz olarak karşılamaktadır. Ancak, yazılım yaşam döngüsü (SDLC) prensipleri gereği, her yazılım projesi sürekli geliştirmeye açıktır. Aşağıda, projenin sonraki sürümlerinde (v2.0) eklenmesi planlanan özellikler ve mevcut sistemin kısıtları detaylandırılmıştır.

4.1. Aerodinamik Sürtünme Kuvveti (Hava Direnci)

Mevcut simülasyon modelinde, hesaplamalar "ideal vakum ortamı" varsayımı ile yapılmıştır. Yani, cisimlere etki eden tek kuvvetin kütleçekimi olduğu kabul edilmiştir. Ancak gerçek fiziksel koşullarda, özellikle Venüs veya Dünya gibi yoğun atmosfere sahip gezegenlerde, cisim hareket ederken bir sürtünme kuvvetine ($F_{sür}$) maruz kalır.

Gelecek geliştirmelerde, aşağıdaki "Sürüklenme Denklemi" (Drag Equation) projeye entegre edilebilir:

$$F_d = \frac{1}{2}\rho v^2 C_d A$$

Burada gezegenlerin atmosfer yoğunlukları (ρ) sisteme eklenerek, serbest düşme deneyinde cismin ulaşabileceği limit hız (terminal velocity) hesaplanabilir. Bu sayede simülasyon, Mars'taki ince atmosfer ile Jüpiter'deki yoğun gaz ortamı arasındaki farkı daha gerçekçi bir şekilde yansıtabilecektir.

4.2. Grafiksel Kullanıcı Arayüzü (GUI) ve Veri Görselleştirme

Mevcut uygulama, C dilinin standart kütüphaneleri kullanılarak Konsol (Command Line Interface - CLI) tabanlı olarak geliştirilmiştir. Bu yapı, geliştirici açısından performanslı ve hafif olsa da son kullanıcı deneyimi (UX) açısından sınırlıdır.

İlerleyen aşamalarda, projenin sadece metin tabanlı çıktılar vermek yerine, sonuçları grafiksel olarak sunması hedeflenmektedir. Bu amaçla:

- **GTK** veya **Qt** kütüphaneleri kullanılarak pencereli bir arayüz tasarlanabilir.
- **OpenGL** veya **SDL** kütüphaneleri kullanılarak, "Serbest Düşme" veya "Eğik Atış" deneyleri sırasında cismin hareketi animasyon olarak (2D veya 3D) ekrana çizdirilebilir.
- Zaman-Konum veya Hız-Zaman grafikleri anlık olarak çizdirilerek kullanıcının veriyi daha iyi analiz etmesi sağlanabilir.

4.3. Veri Kalıcılığı ve Raporlama (File I/O)

Şu anki sürümde, deney sonuçları "volatile" (uçucu) bellek üzerinde tutulmakta ve sadece ekrana yazdırılmaktadır. Program kapatıldığında yapılan hesaplamalar kaybolmaktadır.

Bilimsel çalışmalarda verilerin saklanması kritik önem taşıdığından, projeye **Dosya Giriş/Çıkış (File I/O)** yetenekleri kazandırılmalıdır. Sonuçların:

1. **.TXT formatında:** İnsan tarafından okunabilir basit raporlar halinde,
2. **.CSV (Virgülle Ayrılmış Değerler) formatında:** Microsoft Excel veya MATLAB gibi harici analiz araçlarına aktarılabilecek şekilde,
3. .JSON formatında: Web tabanlı uygulamalarla veri alışverişi yapabilecek yapıda

kaydedilmesi (Export) özelliği eklenmelidir. Bu geliştirme için C dilinin <stdio.h> kütüphanesindeki fprintf ve fopen fonksiyonları kullanılarak bir "Loglama Modülü" yazılabilir.

4.4. Dinamik Gezegen Yönetimi

Mevcut sistemde gezegen verileri (isimler ve g değerleri) config.h dosyasında statik diziler (static arrays) olarak tanımlanmıştır. Yeni bir gök cismi (örneğin Ay, Titan uydusu veya cüce gezegen Plüton) eklemek için kaynak kodun değiştirilmesi ve programın yeniden derlenmesi gerekmektedir.

Geliştirme olarak, programın açılışta harici bir konfigürasyon dosyasından (örneğin planets.conf) gezegen verilerini okuması sağlanabilir. "Dinamik Bellek Yönetimi" (malloc, realloc) kullanılarak, kullanıcı çalışma zamanında (runtime) dilediği kadar yeni gezegen ekleyip çıkarabilir hale getirilebilir.

5. SONUÇ

Bu proje çalışması sonucunda, teorik olarak öğrenilen C programlama dili temelleri, pratik ve kapsamlı bir uygulama üzerinde başarıyla test edilmiştir. Özellikle "Algoritmalar ve Programlama" dersinin en kritik konularından biri olan **işaretçiler (pointers)** ve bellek yönetimi konusunda derinlemesine bir yetkinlik kazanılmıştır. Dizilere indeks yerine bellek adresleri üzerinden erişim sağlanması (pointer aritmetiği), bilgisayarın çalışma mantığının ve bellek organizasyonunun daha iyi anlaşılmasını sağlamıştır.

Projenin çıktıları incelendiğinde, fizik kurallarının evrenselliği ile yerçekimi ivmesinin değişkenliği arasındaki ilişki net bir şekilde görülmektedir. Örneğin, Jüpiter'deki (24.79 m/s^2) bir cismin ağırlığı ile Mars'taki (3.71 m/s^2) ağırlığı arasındaki büyük fark, simülasyon sayesinde somut verilerle ortaya konulmuştur. Hazırlanan 9 farklı deney (Serbest Düşme, Sarkaç, Basit Harmonik Hareket vb.), programın sadece matematiksel bir hesap makinesi olmadığını, aynı zamanda bir fizik laboratuvarı işlevi gördüğünü kanıtlamaktadır.

Yazılım mimarisi açısından bakıldığında, kodun main.c, utils.h ve config.h şeklinde modüler parçalara ayrılması, "Temiz Kod" (Clean Code) prensiplerinin uygulandığını göstermektedir. Bu yapı sayesinde, ileride Güneş Sistemi'ne yeni bir gök cismi (örneğin Ay veya Plüton) eklenmek istendiğinde, ana algoritmayı bozmadan sadece konfigürasyon dosyasını güncellemek yeterli olacaktır. Sonuç olarak, bu proje hem

mühendislik hesaplamaları hem de yazılım geliştirme disiplini açısından hedeflenen tüm kazanımları başarıyla sağlamıştır.