“Üç Cisim Problemi”

Problemin Tanımı

Üç cisim problemi, klasik fizikte birbirlerinin kütle çekimine maruz kalan üç noktasal kütlenin izleyecekleri yörüngelerin hesabını ele alır. Problemin iki cisimli daha basit bir sürümünde aranan yörüngeler temel cebirsel fonksiyonlarla ifade edilebilir. Ancak cisim sayısı üçe çıktığında cisimlerin izleyeceği yörüngeleri genel bir cebirsel formüle oturtmak olanaksızdır.

Bununla birlikte probleme nümerik bir çözüm getirilebilir. Kütleleri, başlangıç konumları ve hızları verilmiş noktasal cisimlerin anlık ivmelenmeleri **Newton’ın Kütle Çekim Yasası** (Eş.1) ve **Newton’ın 2. Hareket Yasası** (Eş.2) uyarınca belirlenebilir.

$$\vec{\boldsymbol{F}}_A^B = -G\,\frac{m_A\,m_B}{|\vec{\boldsymbol{x}}_A - \vec{\boldsymbol{x}}_B|^3}\,(\vec{\boldsymbol{x}}_A - \vec{\boldsymbol{x}}_B) \tag{1}$$

$$\vec{\boldsymbol{a}}_A = \frac{\vec{\boldsymbol{F}}_A^{\Sigma}}{m_A} \tag{2}$$

Eş.1‘de $\boldsymbol{F}_A^B$ vektörü, A noktasal cismine B noktasal cisminin uyguladığı kütle çekim kuvvetini ifade eder. $\boldsymbol{x}_A$ ve $\boldsymbol{x}_B$ vektörleri sırasıyla A ve B cisimlerinin konumlarını, m_A ve m_B değerleri ise bu cisimlerin kütlelerini ifade eder. G kütle çekim sabiti olup hesaplamayı basitleştirmek adına seçilecek uygun bir ölçüm sistemine göre $G = 1$ olduğu kabul edilebilir.

Eş.2’de $\boldsymbol{a}_A(t) = \boldsymbol{x}_A''(t)$, A cisminin anlık ivmelenmesi olup $\boldsymbol{x}_A(t)$ konumunun zamana göre ikinci dereceden türevi olarak tanımlanmıştır. $\boldsymbol{F}_A^{\Sigma} = \boldsymbol{F}_A^B + \boldsymbol{F}_A^C + \cdots$ ise A cisminin maruz kaldığı kuvvetlerin (diğer $B, C, \ldots$ cisimlerinin uyguladığı kütle çekim kuvvetlerinin) bileşkesini ifade eder.

Yukarıda verilen Eş.1 ve Eş.2 üzerinden cismin anlık ivmesi hesaplanabilir. İvme, hızın zamana göre türevidir ve dolayısıyla hızın anlık değişimini ifade eder. Anlık hız ve ivme bilindiğine göre bu an için yazılabilecek Taylor serisini ilk iki terime kadar götürerek Eş.3’te görüldüğü gibi hızın bir sonraki değeri yaklaşık olarak hesaplanabilir.

$$\vec{\boldsymbol{v}}(t + \Delta t) \cong \vec{\boldsymbol{v}}(t) + \vec{\boldsymbol{a}}(t)\,\Delta t \tag{3}$$

$$\vec{\boldsymbol{x}}(t + \Delta t) \cong \vec{\boldsymbol{x}}(t) + \vec{\boldsymbol{v}}(t)\,\Delta t \tag{4}$$

Burada $\boldsymbol{v}(t)$ ve $\boldsymbol{a}(t)$ cismin t anındaki anlık hız ve ivme vektörlerini ifade ederken Δt şimdiki anla (t) hesaplanmak istenen bir sonraki an $(t + \Delta t)$ arasında geçen süredir. Benzer biçimde hız, konumun zamana göre türevi olduğundan yine Taylor serisini ilk iki terime kadar götürerek Eş.4’te görüldüğü gibi konum vektörünün bir sonraki değeri yaklaşık olarak hesaplanabilir. Burada $\boldsymbol{x}(t)$ cismin t anındaki anlık konumunu ifade etmektedir.

Yukarıdaki kuramsal çerçeveye dahil edilebilecek bir başka fiziksel yapı da roketlerdir. Roketler de bütün fiziksel cisimler gibi yukarıdaki yasalara tabidir. Ancak eylemsiz gök cisimlerinin aksine bir roket geriye doğru püskürttüğü çıktı gazlarıyla ek bir itme kuvveti yaratır. Bu ek kuvvet Tsiolkovsky roket denklemleriyle (Eş.5 ve Eş.6) hesaplanabilir.

$$\vec{\boldsymbol{F}}_R = -w_P \vec{\boldsymbol{v}}_P \tag{5}$$

$$m_R(t + \Delta t) = m_R(t) - w_P \, \Delta t \tag{6}$$

Eş.5'te $\boldsymbol{F}_R$ roket motorunun sağladığı ek itki kuvvetini gösterirken $\boldsymbol{v}_P$ roketin püskürttüğü çıktı gazının rokete göreli hızının vektörel ifadesidir. w_P ise birim zamanda püskürtülen çıktı gazının kütlesidir. Roket çalıştığı sürece $\boldsymbol{v}_P$ hızının ve w_P oranının sabit olduğu varsayılır. Eş.6'da $m_R(t)$ roketin anlık toplam kütlesidir ve roketin çıktı gazını püskürttüğü her Δt süresince düzenli olarak azalmaktadır.

Roket için ivme ve hız hesaplaması yapılırken Eş.5'te verilen ek itki kuvveti, roketin maruz kaldığı kütle çekim kuvvetlerine eklenmelidir. Bileşke kuvvet belirlendikten sonra Eş.2 üzerinden anlık ivme bulunabilir. Ancak Eş.2'de kullanılacak kütlenin de anlık bir değer olduğu, roket kütlesinin sürekli azaldığı unutulmamalıdır. Bu nedenle bir sonraki anın $(t + \Delta t)$ roket kütlesi Eş.6 uyarınca güncellenmelidir. Elbette kütlenin sıfırın altına düşemeyeceği de göz önüne alınmalıdır.

Program İsterleri

Yukarıda özetlenen fiziksel modelin benzetimini nümerik olarak yürütecek bir programı C++ dilinde yazmanız istenmektedir. Problem 2 boyutlu düzlemde sınırlandırılmıştır. Benzetimi nümerik olarak yürütürken seçilecek Δt zaman adımları Eş.3 ve Eş.4'deki yaklaşık hesaplamayı yeterince doğru kılacak kadar küçük ($\Delta t^2 \ll 1$) ancak hesaplama süresini çok fazla uzatmayacak kadar büyük seçilmelidir. Bu değer programın genel bir değişmezi olarak tanıtılabilir.

Nesne yönelimli programlama anlayışı uyarınca verilen eşitliklerdeki cebirsel hesaplamaları daha kolay uygulayabilmek adına 2 boyutlu vektörleri içeren bir sınıf da öncesinde tanıtılmalıdır. Sınıfın tüm verileri özel olmalıdır. Ancak gerekli sınıflara bu verilere erişme yetkisi verilebilir. Sınıf için gerekli tüm operatörler aşırı yüklenerek tanımlanmalıdır.

Ardından modeldeki fiziksel cisimleri gösterecek bir sınıf tanıtılmalıdır. Sınıfa cismin kütle, hız ve konum gibi vektörel değerlerini tutmak üzere daha önce tanıttığınız sınıftan üye nesneler eklenmelidir. Cisimlerin maruz kaldıkları bileşke kuvveti parametre olarak alıp buradan cismin anlık hızı ve konumunu güncelleyen bir üye fonksiyon eklenmelidir.

Roketleri göstermek içinse cisim sınıfından yeni bir sınıf türetilebilir. Bu yeni sınıfa roketin $\boldsymbol{v}_P$ püskürtme hızı ve w_P püskürtme oranını belirleyen üyeler eklenmelidir. Temel sınıfa ait anlık hız ve konumu güncelleyen üye fonksiyon, roketler için türetilen sınıfta bastırılmalı; roketin ek itki kuvvetini de hesaba ekleyen ve ayrıca anlık kütleyi de güncelleyen bir yeni sürümü eklenmelidir.

Sisteme eklenecek sınırsız sayıda cisim ve roketi tutacak bir bağlı listeye gereksinim olacaktır. Listenin her bir düğümü cisim ya da roket nesnesinin adresini bir üye göstericide taşıyabilir. Listeye sınırsızca ve rastgele sırada cisim ve roket sınıfından nesneler eklenebilmelidir. (Not: C++ standart kütüphanesince sağlanan `vector<>` ya da `list<>` şablonlarını kullanmayınız.)

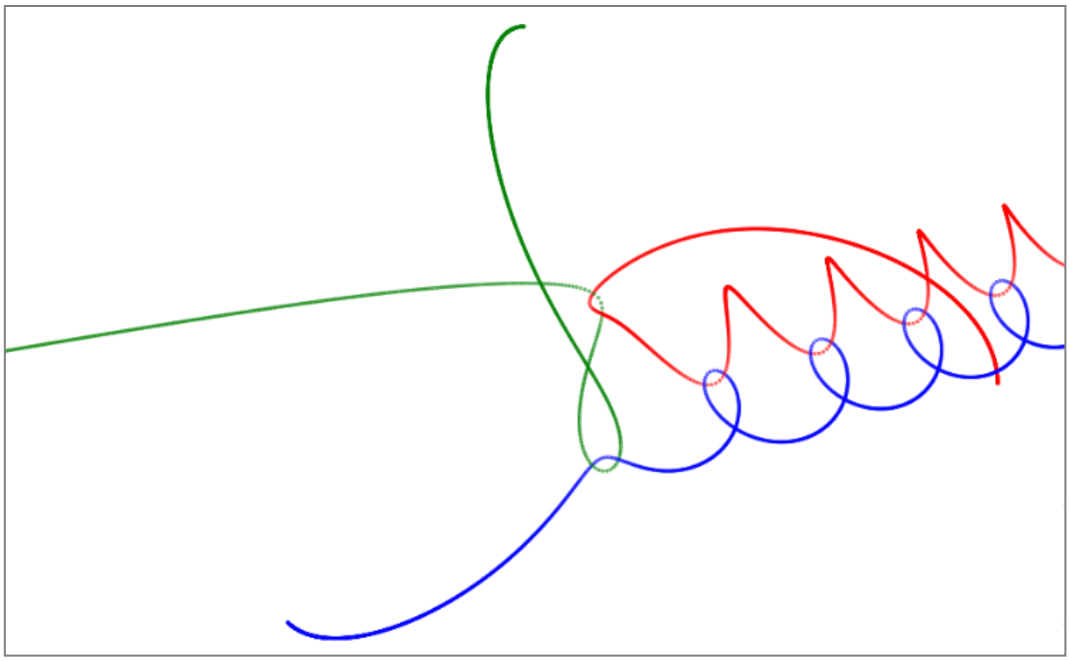
Eklenen cisim ve roketlerin karşılıklı etkileşimlerini yönetecek bir sınıfa daha gereksinim olacaktır. Bu sınıf tuttuğu cisim ve roketlerin birbirlerine uyguladıkları kütle çekim kuvvetlerini hesaplamalıdır. Ardından her bir cisim ve roketin maruz kaldığı bileşke kuvveti üzerinden kendi hız ve konumunu güncellemesini sağlamalıdır. Bu sınıf cisimlerin konum verilerine de erişebilmelidir.

Son olarak cisim ve roketlerin hesaplanan konumları üzerinden izledikleri yörüngeler farklı renklerde çizdirilmelidir. Çizim işlemi için `canvas.cpp` kütüphanesi kullanılabilir. Gerekli sınıflar tanıtıldıktan ve üye fonksiyonlar tanımlandıktan sonra ana fonksiyonda bu sınıfların örnek bir programda kullanımı ve buradan hesaplanan yörüngelerin çizimi aşağıda verilmiştir:

Örnek Program

```
01: #include "gravity.cpp"
02: #include "canvas.cpp"
03:
04: int main()
05: {
06:     canvas graphic("gravity");
07:     graphic.startDoc();
08:     graphic.drawFrame();
09:
10:     /* create three bodies with distinct mass and initial position & velocity */
11:     body m1(150, vector(400,0), vector(0,0.2));
12:     body m2(70, vector(0,300), vector(-0.15,0));
13:     body m3(150, vector(-200,-200), vector(0.1,-0.1));
14:
15:     /* create a system and insert the bodies */
16:     universe U;
17:     U.insertBody(m1);
18:     U.insertBody(m2);
19:     U.insertBody(m3);
20:
21:     for(int t=0; t<2000; t++)
22:     {
23:         U.run(timestep*100); // run simulation for a while and draw last position
24:
25:         graphic.drawPoint(U.getPosX(1), U.getPosY(1),"red");
26:         graphic.drawPoint(U.getPosX(2), U.getPosY(2),"green");
27:         graphic.drawPoint(U.getPosX(3), U.getPosY(3),"blue");
28:     }
29:
30:     graphic.finishDoc();
31:     return 0;
32: }
```

Programda tanıtılacak sınıfların tüm verileri özel ya da korunaklı olmalıdır. Programı hız ve bellek kullanımı yönünden iyileştirmek adına fonksiyonlarda nesne aktarımları nesnelerin ikinci bir kopyası oluşturulmadan gerçekleştirilmelidir. Program tamamlanmadan önce geride bağlı liste de dahil olmak üzere bellekten silinmemiş hiçbir veri bırakılmadığından ve bellek sızıntısına yol açılmadığından emin olunuz. Bunun için gerekli ek fonksiyonları da sınıflara ekleyiniz.



Şekil 1. Örnek programın çıktısında çizdirilen yörüngeler

Projeden beklenen çıktılar aşağıda sıralanmıştır:

- ✓ **Program Dosyaları** (`.cpp`, `.h`)
- ✓ **Proje Raporu** (`.pdf`)
  - Hedeflenen programın kısa özeti
  - Nesne yönelimli programlama anlayışı çerçevesinde program yazımının parçalara ayrılması ve iki kişiden oluşacak ekip içinde paylaşımı
  - Nesne yönelimli programlama anlayışı çerçevesinde tanıtılan ve tanımlanan tüm sınıf, nesne, değişken, üye değişken, üye fonksiyon vb. öğelerin tanımları, kullanım amaçları ve yerleri
  - Programın akışının ayrıntılı açıklaması
  - Programın çalışma hızını belirleyen etmenlerin incelenmesi ve yapılan iyileştirmeler
  - Programın bellek yönetimi yönüyle incelenmesi ve yapılan iyileştirmeler
  - Programda kullanılan sınıflar arasındaki kalıtımın incelenmesi
  - Programın çokbiçimlilik yönünden incelenmesi
  - Programda kullanılan veri yapılarının incelenmesi
  - Farklı sayıda ve başlangıç değerleri ile sisteme eklenecek cisim ve roketlerin izledikleri yörüngelere ait çizimler
  - Programa eklenen yeni özellikler (Örneğin cisim ve roketlerin terminal ekranı üzerinden bir arayüzle girilebilmesi, cisim sınıfından türetilebilecek yeni sınıflar vb.)
  - Benzetimden elde edilebilecek fiziksel gözlemler (Örneğin sistemdeki tüm cisimlerin ağırlık merkezlerinin hareketi, momentumun korunumu vb.)
- ✓ **Farklı Değerler ile Yörünge Çizimleri** (`.html`)