T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

MEKANİK FİZİK HAREKETLERİNİN 3 BOYUTLU ORTAMDA SİMULASYONU

YÜKSEK LİSANS TEZİ

Hazırlayan
Hüseyin ADALAR

Tez Danışmanı
Yrd. Doç. Dr. Murat BEKEN

İstanbul - 2009

ÖNSÖZ

Projenin geliştirilmesi sırasında bana güveni ve değerli rehberliği için danışman öğretmenim Yrd. Doç. Dr. Murat Beken'e teşekkürlerimi sunarım.

Ayrıca her zaman için destek ve sabırlarını esirgemeyen, moral ve motivasyon kaynağı olan aileme de teşekkürlerimi sunarım.

İÇİNDEKİLER

T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ
YÜKSEK LİSANS TEZİ

Mekanik Fizik Hareketlerinin 3 Boyutlu Ortamda Simülasyonu

Hazırlayan
Hüseyin ADALAR

Tez Danışmanı
Yrd. Doç. Dr Murat Beken

Ocak 2009

ÖZET

Simülasyon, günümüzde pozitif bilimlerde çok önemli bir yere sahip, uygulamadır. Birçok sistemin geliştirilmesinde ve ilerletilmesinde simülasyon metoduna sık sık başvurulur. Simülasyon metodu bir yandan yeni sistemlerin geliştirilmesinde başrol oynarken; diğer yandan zaman içinde kendini de geliştirmek için çeşitli metotlar kullanmıştır. Bu metotlar için de 3D (3 boyut) simülasyonlar en önemlilerinden biridir. Özellikle gerçek dünyanın ve geleceğe yönelik çıkarımların planlanmasında 3D simülasyonlara sıklıkla başvurulmaktadır.

Bu projede de 3D simülasyon metodunu baz alarak, mekanik fizik (newton fiziği) konularından 5 tanesinin 3D ortamında simülasyonu gerçekleştirilmiştir. Bu konular, enerji korunumu, kuvvet ve hareket, düzgün doğrusal hareket, basit harmonik hareket (sarkaç), ve eğik atış hareketidir.

Projeyi geliştirme ortamı için java dili seçilmiştir. Java standart edition 1.6 ve java 3d 1.4 kütüphanelerini kullanılmıştır. Fizik hareketleri simüle edilirken, her bir hareketin formüllerinin değişkenleri enteraktif ara yüzle kullanıcı tarafından

girilmiştir. Hareketlerin enerji, hız, ivme ve yer değiştirme grafikleri eş zamanlı olarak gösterilmiştir.

Anahtar kelimeler: java 3d, simülasyo

T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ
YÜKSEK LİSANS TEZİ

3 Dimension Simulation of Mechanic Physics Motions

Prepared
Hüseyin ADALAR

Instructor
Yrd. Doç. Dr Murat Beken

January 2009

ABSTRACT

Simulation is essential in sciences today. Many valuable systems is developed or updated by simulations. During simulations is improving the system, it improves itself. One of the essential method to improve simulations is 3d environment. Especially simulating real world and modeling future ideas, 3d simulations are used.

In this project, 5 of mechanical physics motions are simulatedby 3d environment. This motions are, force,energy saving, uniform circular motion, pendulum motion and projectile motion.

Java language is preffered to develop this project. Java se 1.6 and java 3d 1.4 libraries are used. All variables for motions formulas are parametric, and taken by user with graphical user interface. Motions graphs are also drawn on the screen.

Key words: java 3d, simulation

ŞEKİLLER

TABLOLAR

1. GİRİŞ

1.1 Java

Java, Sun microsystems tarafından geliştirilmeye başlanmış, gerçek nesneye yönelik (object oriented), platform bağımsız, yüksek performanslı, çok işlevli, yüksek seviye, interpreted (adım adım işletilen) bir dildir[1].

Java ilk olarak oak (sun tarafından geliştirilmiş, java'nın atası olan nesneye yönelik bir dil) adı ile küçük cihazlarda bütünleşik (entegre) kullanılmak için tasarlanmış ortak bir platform dili olarak düşünülmüştür. Fakat platform bağımsız özelliği, nesneye yönelik oluşu, web (internet sayfalarının genel adı) ve mobil teknolojilere yönelik geniş kütüphanesi ile kurumsal ve akademik alanda yaygın olarak kullanılan bir dil haline gelmiştir.

1.1.1 Java'nın Tarihsel Gelişimi

Java, ilk olarak 1995 yılında duyurulmuş olsa da kökenleri 1991 yılına kadar dayanan bir dildir. Java'nın ilk sürümü olan Java 1.0 (1995) Java Platform 1 olarak adlandırıldı[2]. Tasarlanma amacına uygun olarak küçük boyutlu ve kısıtlı özelliklere sahipti. Daha sonra platformun gücü gözlendi ve tasarımında büyük değişiklikler ve eklemeler yapıldı. Bu büyük değişikliklerden dolayı geliştirilen yeni platforma Java Platform 2 adı verildi. Java 1.5 tiger versiyonu (2004) ile birlikte Java 5 platformu olarak adlandırıldı. Çekirdek yapısının yanı sıra, kurumsal web uygulamaları için J2EE (java'nın kurumsal web uygulamaları için geliştirilmiş versiyonu), mobil uygulamaları için J2ME (java'nın mobil uygulamalar için geliştirilmiş teknolojisi) versiyonları da zaman içinde yayınlandı. Aşağıda Java dilinin tarihsel gelişiminin tablosu sunulmuştur.

---

[1] : Harvey M.Deitel, Paul J.Deitel, Java How To Progam (5th Ed.), Deitel & Deitel

[2] Cay S.Horstmann, Gary Cornell, Core Java Volume 1 Fundementals, Prentice Hall

1991	The Green Project başladı. "Biosphere 2" projesi başladı.
1992	Oak dili yayınlandı.
1995	Java dili resmi olarak duyuruldu. Oak dilinin adı Java olarak değiştirildi.
1996	Geleneksel JavaOne konferanslarının birincisi yapıldı. JDK 1.0 yayınlandı.
1997	Jdk 1.1 yayınlandı ve 3 hafta içinde 220.000 kez internet üzerinden indirildi JavaOne konferansına 8000 kişi katıldı. Java Card 2.0 platformu duyuruldu.
1998	Java Card teknolojisi visa kartlarda kullanılmaya başlandı. The Java Community Process (JCP) programı resmi hale getirildi.
1999	Java 2 platformunun kaynak kodu yayınlandı. JavaOne 20.000 katılımcı ile gerçekleştirildi. J2EE beta yazılımı yayınlandı
2000	Dünya üzerinde 400 ün üzerinde Java geliştirme grubu kuruldu. Java Developer Connection programı 1.5 milyon üyeye ulaştı.
2001	JavaOne konferansı ilk olarak ABD dışında Japonya'da düzenlendi. J2EE teknolojisi 1 milyondan fazla download edildi.
2002	J2EE teknolojisi 2 milyondan fazla download edildi. Web uygulamaları teknolojilerinde J2EE'nin payı %78'e yükseldi.
2003	Java teknolojisi 550 milyon kişisel bilgisayarda kullanılır hale geldi. Profesyonel yazılım uzmanlarının %75 oranının birinci tercihi Java dili oldu.
2004	Java 2 Platformu, Java 5 (Tiger) yayınlandı.
2005	Java teknolojisi 10. yaşını kutladı. Dünya üzerinde Java kullanan yazılımcılarına sayısı 4.5 milyona ulaştı. 2.5 milyon üzerindeki Java sistemi çalışır hale geldi.
2006	Java EE ve Java ME teknolojileri açık kaynak haline getirildi. Netbeans 5.0 Java geliştirme ortamı yayınlandı.

Tablo 1.1: java'nın tarihi gelişimi

1.2 Java'nın Kullanım Alanları

Nesneye yönelik yapısı ve geniş kütüphanesi ile Java dünya üzerinde çok geniş kullanım alanınsa sahip bir platformdur.

Web Sayfaları: Java teknolojilerinden olan applet ve jsp teknolojisi web sayfalarında yaygın olarak kullanılırlar. Applet teknolojisi web sayfası içerisine küçük uygulamalar gömmek için uygun bir teknolojidir. Web sayfalarında yer alan oyun ve anlık mesaj programlarının büyük kısmı appletler ile geliştirilmiştir.

Jsp (dinamik web sayfası yapmaya yarayışlı dil) teknolojisi de dinamik web sayfalarının geliştirilmesi için kullanılan teknolojidir. Fakat günümüzde artık jsp kendi başına kullanılmamakta, J2EE içerisinde yer almaktadır.

Web Uygulamaları: Java platformunun en önemli teknolojilerinden olan J2EE (Java enterprise edition) teknolojisi, web tabanlı büyük uygulamaların geliştirilmesine olanak sağlamıştır. Yıllar içerisinde gelişen tasarım kalıplarının da desteği ile oldukça yaygınlaşan J2EE teknolojisi web uygulamaları alanında en yaygın kullanıma sahip hale gelmiştir. Dünya üzerindeki lisanslama yoluyla satılan pek çok büyük web tabanlı uygulama bu teknoloji ile geliştirilmiştir.

Mobil Uygulamaları: Java'nın kullanım alanlarında biri de el cihazları veya çeşitli makinelerdir. Bu cihazlar da Java'nın J2ME (Java micro edition) adlı teknolojisi çalışır. Standart Java'dan farklı olan bu versiyon, bazı ek sınıflar içerdiği gibi bazı standart kütüphaneleri de içermez. Java'yı destekleyen cep telefonları, buzdolapları, arabalar J2ME teknolojisi ile geliştirişmiş uygulamaları kullanmaktadırlar. Yakın bir gelecekte J2ME kullanım alanının daha da yaygınlaşacağını düşünülmektedir.

1.3 Java3d

Java3d hakkında bilgi verilmesinden önce, yazılım dünyasında 3D'nin geçmişinden bahsetmek ve diğer 3D kütüphaneleri tanıtmak faydalı olacaktır.

1.3.1 3D Grafik Api'lerine Genel Bakış

Dünya üzerindeki en önemli 3D api'leri,

Core Api: Siggraf tarafından 1970'lerde çıkarılmıştır.
Phings 3Dgks: 1980 başlarına doğru çıkan ikinci nesil api'ydi. SUN, Apollo, Mosoic gibi şirketler tarafından kullanılmıştır.

OpenGL (Open Graphics Library): Sierra tarafından geliştirilmiş, 3D uygulamalar yazmak için dilden ve platformdan bağımsız standart bir kütüphanedir. Üç boyutlu, hatta iki boyutlu karmaşık şekiller çizmek için uygundur. Günümüz oyunlarının çoğu OpenGL tabanlıdır.

DirectX: OpenGL'e alternatif olarak Microsoft tarafından geliştirilmiş üç boyutlu grafik ve çoklu ortam kütüphanesidir. Günümüz 3D uygulamaları ve özellikle oyunlar DirectX destekli geliştirilmektedir.

Java3D: 1990 sonlarına doğru Sun şirketi tarafından geliştirildi. Java3D directmodel (HP), scenegraph optimizertoolkit (SGI), fahrenheit (SGI and Microsoft) programlara rağmen yeni scenegraphı, büyük modellerde optimizasyon kapasitesi ve birçok yenilikler sayesinde yeni nesil bir api standardının habercisi olmuştur.

1.3.2 Dördüncü Nesil 3D Grafik Api'si: Java 3D

Java3D scenegraph (3D uygulamalarda kullanılan her parçanın coğrafik ve/veya mantıksal ilişkilerinin belirlendiği hiyerarşik veri yapısı) yapısında nesneleri tutan bir modele sahiptir. Scenegraph yapısında olması, daha sağlam ve daha doğru sonuçlar üretilmesine, tutulan nesneler üzerinde daha kolay işlem yapılabilmesine olanak sağladığından güncelleme ve render (3D modelden gerçek şekillerin elde edilmesi işlemi) işlemleri daha hızlı gerçekleşir[3].

Java3D, kendisinden önceki apilerde kullanılan backface removal, bsp trees algoritmalarını api'ye dahil edip, ayrıca yeni algoritmalar da kullanarak, tümü düzenli ve üzerinde kolay işlem yapılabilir bir ortam yaratmıştır. Bu yeni mimari

[3] : Sowizral Henry, Rushforth Kevin, The Java 3D API Specification (2nd Ed.), Addison Wesley

sayesinde process (işlem), birbirinden bağımsız halde parçalara bölünerek threadler (render, ağaç üzerinde gezme, optimizasyon, behavour) oluşturarak çalıştırılır. Bu sayede multiprocessing (aynı anda birçok işlem yapılması) yapılmış olur.

Ayrıca hiyerarşik 3D ağaç yapısı modelinde yolunu kaybetmiş nesnelerin silinmesi ve garbage collection (bellek temizleme işlemi) işlemleri api tarafından üstlenilir.

Scenegraph aynı objeye birden fazla tanım ile ulaşmaya olanak sağlayarak bazı mantıksal ve stratejik durumlarda büyük yarar sağlar. Örnek olarak arabanın dört tekerleğini ayrı ayrı oluşturmak yerine, bir tanesini oluşturup, diğer 4' ünü aynı nesneye farklı referanslar yapmayı sağlar ve her bir tekerleği yeniden oluşturma yükünden kurtarır.

Yeni algoritma eklenmesiyle renderleme işleminde pipelinenin (bilgisayar bilimi dünyasında veri işleme modellerinden bir tanesi) ve disk ile bellek arasındaki veri aktarımının başarılı bir şekilde yapılmasına olanak sağlar. Bu şekilde grafik performansı artmış olur, bellek ve disk kullanımı optimize edilir.

Bu yeni eklenen optimizasyon kapasitesi birbirinden ayrı işlemler şeklinde renderleme ve pipelinenin programlarda nasıl çalıştığı konusunda bilgiye sahip olmak zordur. Bu sebeple 4.nesil grafik api'sinin 3d grafik kullanıcısının üzerinden optimizasyon ve rendering gibi ağır konuları alarak arka tarafta en iyi şekilde kendisinin düzenlemesi sonucu, programcıya da sadece programda neler gözükeceği ve ne yapacağı konuları kalmıştır.

1.3.3 3D Api'lerinin Karşılaştırılması

OpenGL makine dili şeklinde tasarlanmıştır, fakat Java3D yüksek seviyeli bir dildir[4]. Yüksek seviyeli dil olması sebebi ile OpenGL'den temel bazı farklılıkları vardır.

OpenGL kullanıcıları karmaşık scenegraphlar oluşturabilir ve bunun üzerinde nasıl gezilip render işlemi yapılacağına karar verebilirler. Fakat çalışma zamanı sırasında scenegraphlar üzerinde oynama isteği belli kurallar çerçevesinde yapılabilir, ağaç yapısının bir düğümünde (node) yapılan değişiklik tüm düğümü etkiler ve neticede scenegraphta sürekli yeni yapıya göre ayarlamalar gerekir. Bu

[4] : Gene Davis, Learning Java Binding for OpenGL, AuthorHouse

uygulama bir uzmana büyük bir avantaj sağlarken, normal kullanıcılar için başa çıkılması zor bir durumdur.

Java3D normal kullanıcıyla uzman arasında bir denge kurmaya çalışan bir apidir. Kullanıcıyı birçok ayrıntıdan soyutlayarak üzerinde daha rahat bir şekilde çalışma olanağı sağlar.

1.3.4 Java3D'nin Kullanım Alanları

Web: Web dünyasında vrml (virtual reality modelling language: geometrik formüller yazılarak 3 boyutlu nesneler çizilebilen yazım dili) ile birleşerek sanal gerçeklik projelerinde kullanılmaktadır

Oyun: İnternet üzerinde 3D oyunların hazırlanmasında temel seviyede kullanılmaktadır.

Data Visualization (Veri görselliği) **:** yüksek seviyeli veri aktarımlarının görselleştirilmesi, internet üzerinde uçak ve otomobil simülasyonlarının kullanılabilmesi, astronomi ve biyokimyasal olaylarının incelenebilmesi konularında kullanılmaktadır.

1.4 Projenin Amacı

1.4.1 Simülasyonun Tanımı

Bu projede, günümüz teknolojilerinin ve bilimin gelişiminde, 3 boyutlu bilgisayarlı simülasyonun ne kadar önemli olduğu gösterilmek istenmiştir. Bu önemin detaylandırılmasından önce, simülasyon kavramını açmak gereklidir.

Simülasyon için birkaç tanım verilebilir.

1. Bir sistemin simülasyonu, bu sistemi temsil edebilecek bir model oluşturma işlemidir.
2. Simülasyon gerçek sistemin modelinin tasarlanması ve bu model ile sistemin işletilmesi amacına yönelik olarak, sistemin davranışını anlayabilmek veya değişik stratejileri değerlendirebilmek için deneyler yürütülmesi sürecidir

Yukarıdaki tanımların ışığında, bilgisayarlı simülasyonun tanımını yapmak istersek:
Simülasyon, teoriksel ya da gerçek fiziksel bir sisteme ait neden sonuç ilişkilerinin bir bilgisayar modeline yansıtılmasıyla, değişik koşullar altında gerçek sisteme ait davranışların bilgisayar modelinde izlenmesini sağlayan bir modelleme tekniğidir diyebiliriz.

1.4.2 Simülasyonun Kullanım Amaçları

Simülasyon,

- Belirli kararların sonuçlarını ve gidişatlarını tahmin etmekte
- Gözlemlenen sonuçların sebeplerini belirlemede
- Yatırım yapmadan önce problem alanlarını belirlemede
- Değişikliklerin etkilerini ortaya çıkarmada
- Bütün sistem değişkenlerinin bulunmasını sağlamada
- Fikirleri değerlendirmede ve verimsizlikleri belirlemede
- Yeni fikir geliştirmeyi ve yeni düşünceyi teşvik etmede
- Planların bütünlüğünü ve fizibilitesini test etmede

kullanılır.

3 boyutlu simülasyonlar da, gerçek dünyayı yukarıda belirtilen maddeler çerçevesinde tanımak ve daha iyi hale getirmek için kullanılır. Modern bilimin ve teknolojinin gelişiminde 3 boyutlu simülasyonların büyük önemi vardır. Olası durumlar 3 boyutlu ortamda tasarlanıp simle edilmiş ve çıkan sonuçlar ile önemli buluşlar yapılmıştır. Modern tıp, fizik, biyokimya, genetik vb bilimler 3 boyutlu simülasyonların desteğini yoğun ölçüde görmektedir.

1.4.3 Projenin Amacı

Simülasyonun tanımı kısmında belirtildiği üzere 3 boyutlu simülasyonlar günümüz dünyasında büyük bir önem taşımaktadırlar.

Bu proje, 3 boyutlu simülasyonun önemini ve gerekliliğini kolay anlaşılır bir konu ile açıklamak, bu işlem için java3D platformunun uygun bir platform olduğunu uygulamalı olarak belirtmek ve 3 boyutlu simülasyonlar ile ilgilenmek isteyen araştırmacılara yol göstermek amacını taşımaktadır

2 MATERYAL & METOT

2.1 Projede Kullanılan Materyaller

Projede standart Java için 1.6 versiyonu, 3D için java3D 1.5 versiyonu (www.java.sun.com) kullanılmıştır.

Geliştirme ortamı olarak açık kaynak kodlu ücretsiz bir geliştirme ortamı olan eclipse (www.eclipse.org) kullanılmıştır.

Kaynak kodların UML (kaynak kodların modellenmesi için geliştirişmiş bir model dili) diyagramının çıkarılabilmesi için Altova UModel (www.altova.com) programının 30 günlük deneme sürümü kullanılmıştır.

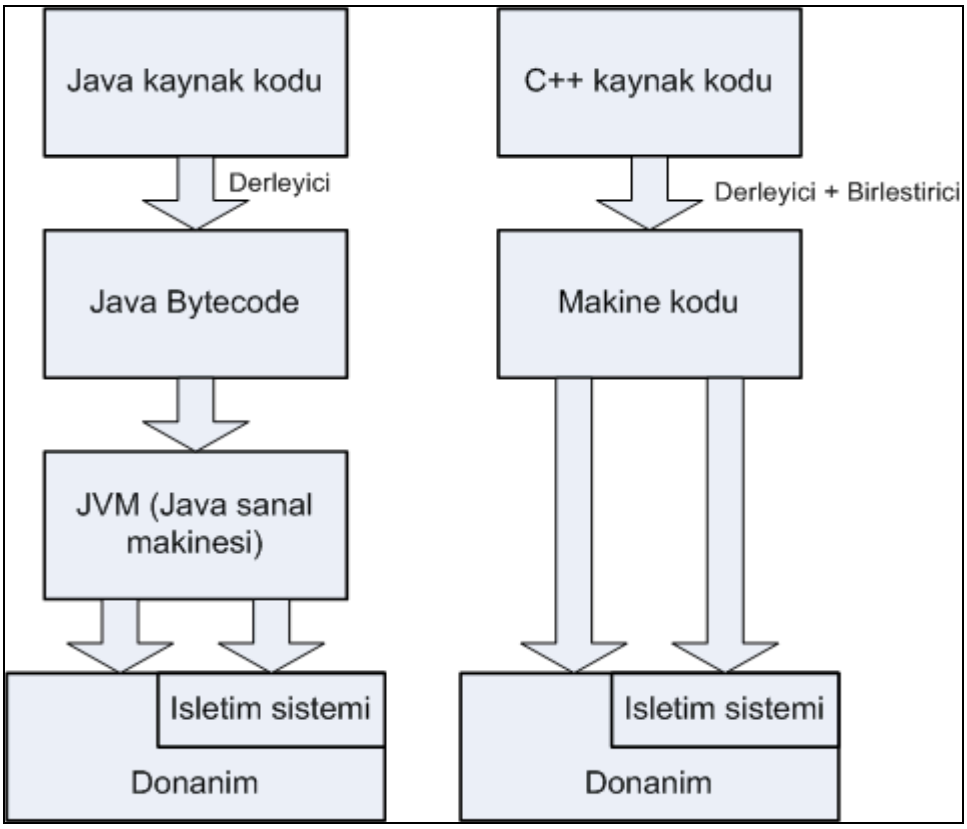
2.2 Java

2.2.1 Java Dilinin Özellikleri

Java dilinin özelliklerini iyi anlamak için, bir Java programının nasıl çalıştığını gözlemlemek gerekir.

Bir Java programı şu şekilde geliştirilir:

- Programcı tarafından Java kodu yazılır.
- Bu kod bir Java derleyicisi ile derlenir. Sonuçta bytekod (java sanal makinesi dili) adi verilen bir tür makine kodu ortaya çıkar. Platform bağımsızlığı bytekod ile sağlanır.
- Bu bytekod Java virtual Machine (Java Sanal Makinesi) tarafından adım adım işletilir ve gerçek makine diline çevrilir.
- Aşağıda Java ve C++ kodunun geçirdiği aşamalar gösterilmiştir.



Şekil 2.1: java ve c++ programlarının aşamaları

2.2.2 Java Teknolojileri

Günümüzde Java, sadece Java dili ile sınırlı kalmayıp birçok teknolojinin birleştiği bir platform haline gelmiştir. Bu teknolojilerin en önemlileri aşağıda listelenmiştir.

JVM: Java sanal makinesi, Java bytekod jvm üzerinde çalıştırılır.

Java Api (api: program geliştirme ara yüzü): Java'nın programcılar için sağladığı yazılım geliştirme ara yüzü. Bütün kütüphaneler api'nin bir parçasıdır.

Garbage collector: programcıyı karmaşık bellek işlemlerinden koruyan, bellek yönetimini üstlenen, geri planda çalışan Java program parçacığı.

Jar (java archive): Java kodlarının çift tıklamayla çalışır hale getirilmesini sağlayan teknoloji.

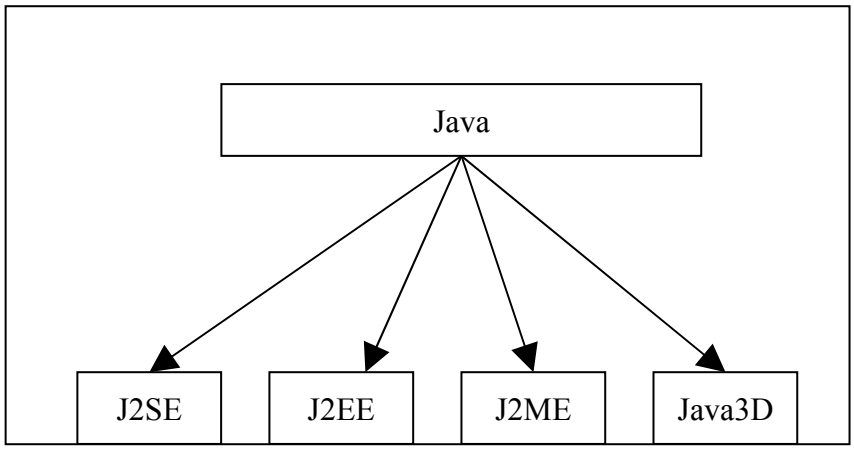
J2EE: kurumsal web uygulamaları için geliştirilmiş, sunucu ve istemci kütüphaneleri barındıran teknoloji

J2ME: mobil uygulamalar için geliştirilmiş teknoloji.

Java3D: yüksek performanslı 3d uygulamaları için geliştirilmiş, temelde OpenGL kütüphanesini kullanan Java teknolojisi

JavaCard: smart kart ve diğer kısıtlı bellek gereksinimi olan cihazlarda kullanılmak üzere geliştirilmiş teknoloji.

Java platformunun genel görüntüsü aşağıdaki gibi düşülebilir.



Şekil 2.2: java teknolojileri hiyerarşisi

2.2.3 Java'nın Avantajları

Java platformu günümüzde çok geniş bir kullanım alanına sahip, programcıya birçok avantajlar sağlayan bir platformdur. Java'nın belli ayırt edici belli başlı avantajlarından bahsetmek gerekirse,

- Platform bağımsız bir dildir. Bir kere yazılıp her platformda çalışabilme özelliğine sahiptir.
- Bellek işlemlerini kendisi üstlendiği için, programcıyı karmaşık bellek işlemlerinden kurtarır.
- Nesneye yönelik (object oriented) bir dildir. Kalıtım (inheritance), kapsüllüme (encapsulation) gibi nesneye yönelik programlamanın temel özelliklerine sahiptir.
- Özellikle kurumsal web uygulamalarında, servlet (sunucu tarafında işletilen sınıflar) mimarisi sayesinde, sunucu tarafında bir kere derlenen kodların bir daha derlenme ihtiyacı hissetmemesi sebebiyle yüksek performansa sahiptir.

- Dünyanın her tarafındaki programcıların da katkısı ile çok geniş bir yazılım kütüphanesine sahiptir ve programcılara çok çeşitli uygulanabilir altyapılar (framework) sunar. Java açık kaynak kodlu ücretsiz binlerce teknoloji ile desteklenir.
- Java aynı anda birden fazla program parçacığını çalıştırmaya izin veren bir yapıya (multi-threaded) sahiptir. Bu özellik gerçek zamanlı uygulamalar için büyük önem taşımaktadır.

2.3 Java 3D

Java platformunun, yüksek performans isteyen üç boyutlu grafiklerde kullanılabilecek kütüphanesidir[5].

OpenGL veya DirectX altyapısını kullanılarak geliştirilmiştir. OpenGL ve DirectX, düşük düzeyli, makine diline daha yakın temel fonksiyonları içerirken; Java3D nesneye yönelik programlamaya uygun bir yazılım altyapısıdır. Java3D'nin bu şekilde nesneye yönelik olmasının sebebi tamamıyla standart Java kütüphanesi şeklinde tasarlarmış olmasındandır.

2.3.1 Java 3D Kütüphaneleri

Java 3D kütüphanesi *java.media.j3d* paketindir. Üç boyutlu nesnelerde çok miktarda vektör ve matris işlemleri olduğu için *javax.vecmath* kütüphanesi de Java3D kütüphanelerine dahildir. Java 3D kütüphanesi çok temel sınıflar içerir, basit bir örnek için bir çok miktarda kod yazmak gerekir. Bu sebeple Sun firması *com.sun.j3d* ile başlayan çok miktarda yardımcı ve faydalı nesne geliştirmiştir. Örneğin Java 3D'de küp yapmaya yarayan bir nesne yoktur, bunun için *com.sun.j3d.utils.geometry.ColorCube* adında, her yüzeyinde değişik bir renk olan bir küp bulunmaktadır.

[5] : Oliver Faulhaber, A Scene-graph Editor for Java 3D, 2005

2.3.2 Canvas3D

Üç boyutlu nesnelerin çizilmesi için bir tuval görevi gören canvas gereklidir. Java 3D kütüphanesi Java'nın standart ara yüz kütüphanesi AWT üzerinde çalışır. Canvas3D sınıfı *java.awt.Canvas* sınıfından türemiştir.

Örneğin,

```
public class ColorCubeTest extends Frame {
  public ColorCubeTest() {
    setLayout(new BorderLayout());
    GraphicsConfiguration config =SimpleUniverse.getPreferredConfiguration();
    Canvas3D canvas = new Canvas3D(config);
    add(BorderLayout.CENTER,canvas);
  }
}
```

şeklindeki kod, sistemin grafik konfigürasyonunu kullanan bir tuval oluşturmakta ve bir pencerenin ortasına yerleştirmektedir.

2.3.3 Universe (Evren)

Java3D'de çizilen bütün nesneler bir evren'de (universe'de) yer alır. Öncelikle tuval (canvas) üzerinde çalışan evren oluşturulur. Evren aslında VirtualUniverse sınıfıyla temsil edilir ancak Sun'ın faydalı sınıfları arasında bu sınıfı genişleten SimpleUniverse sınıfı bulunmaktadır.

Örneğin

```
public class ColorCubeTest extends Frame {
  public ColorCubeTest() {
    // ...
    SimpleUniverse universe = createUniverse(canvas);
    BranchGroup group =createGroup();
    universe.addBranchGraph(group);
```

```
 }
 private SimpleUniverse createUniverse(Canvas3D canvas) {
  SimpleUniverse universe = new SimpleUniverse(canvas);
  ViewingPlatform platform=universe.getViewingPlatform();
  platform.setNominalViewingTransform();
  return universe;
 }
```

kodu bir evren oluşturmakta ve tuvale (canvas) atamaktadır.

2.3.4 Point (Nokta) & Vector (Vektör)

Uzayda bir noktayı göstermek için PointXXX sınıfları bulunmaktadır. Bunlar 2 ve 3 boyutlu ve 4 boyutlu (x,y,z üç boyutu ve homojen koordinatlardaki dördüncü w boyutu) gibi boyut seçeneklerine, integer, float ve double tip seçeneklerine göre çeşitli nokta sınıfları bulunmaktadır. Örneğin Point3d

2.3.5 Behaviour (Davranış)

Normal şartlarda canvas ve evren durağandır. İzleyici gördüğü üç boyutlu nesneler karşısında hareket edemez. Bir kamera hareketi gibi bir işlev görmek için ViewPlatformBehavior sınıfı bulunmaktadır. Behaviour (davranış) sınıfları kullanıcı ile üç boyutlu evren arasında iletişimi sağlar. ViewPlatformBehavior ise, farenin sol tuşuyla şeklin etrafında dönülen, sağ tuşuyla da şeklin karşısında harekete edilen eylemleri destekler. Bir evrene bir davranış eklemek için

```
public class ColorCubeTest extends Frame {
 // ...
 private SimpleUniverse createUniverse(Canvas3D canvas) {
  SimpleUniverse universe = new SimpleUniverse(canvas);
  ViewingPlatform platform=universe.getViewingPlatform();
  platform.setNominalViewingTransform();
  ViewPlatformBehavior behavior=createBehaviour(canvas);
  platform.setViewPlatformBehavior(behavior);
  return universe;
 }
```

```
  private ViewPlatformBehavior createBehaviour(Canvas3D canvas){
    int flags=OrbitBehavior.REVERSE_ALL |OrbitBehavior.STOP_ZOOM;
    OrbitBehavior behavior = new OrbitBehavior(canvas,flags);
    Point3d origin = new Point3d(0.0,0.0,0.0);
    BoundingSphere bounds =new BoundingSphere(origin, 100.0);
    behavior.setSchedulingBounds(bounds);
    return behavior;
  }
}
```

Burada orijin etrafında dönme yapılabilen kullanıcı etkileşimi kodlanmıştır.

2.3.6 BranchGroup (Dal Grubu)

Java3D'de her nesne bir dal grubu (BranchGroup) içerisindedir. Anlam olarak bir veya daha fazla nesne belli bir orijine göre belli bir koordinat sistemine taban alarak konumlandırılmıştır ve bu nesnelerden oluşan dal grubu birlikte hareket eder. Bir üç boyutlu nesne oluşturulduktan sonra bir dal grubu oluşturulur ve nesne o dal grubuna eklenir. O dal grubu da evrene eklenir. Evren > Dal Grubu > Şekil şeklinde bir yapı vardır. Evrende dal grupları, dal gruplarında da şekiller bulunur. Bir dal grubu ötelenir veya döndürülürse o dal grubundaki bütün şekiller ötelenir veya döndürülür. Şekillere örnek olarak ColorCube adında, her yüzeyi değişik renkte olan bir nesne alınabilir. Bu şekli bir dal grubuna eklemek ve bu dal grubuna da bir evrene eklemek için

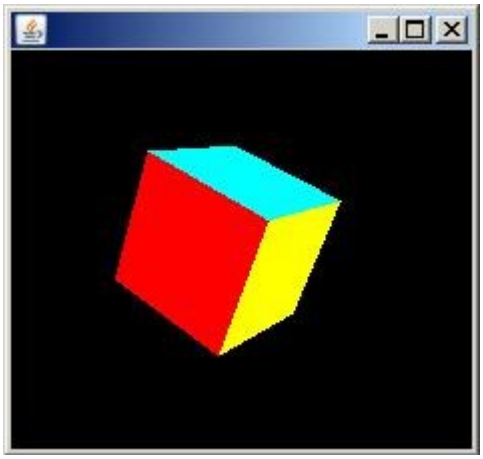
```
public class ColorCubeTest extends Frame {

  public ColorCubeTest() {
    // ...
    BranchGroup group =createGroup();
    universe.addBranchGraph(group);
  }
  private BranchGroup createGroup(){
    BranchGroup group = new BranchGroup();
    group.addChild(new ColorCube(0.3));
```

```
    return group;
  }

}
```

şeklinde bir kod yazılabilir ve aşağıdaki sonucu üretir.



Şekil 2.3: Dal grubu ile oluşturulmuş 3D küp

2.3.7 Light (Işık)

Java3D'de üç boyutlu ortamda ışık kullanılabilir. Işık, PointLight, noktasal (bir noktadan her yöne doğru yayılan); DirectionalLight yani doğrusal (bir noktadan belli bir yöne doğru yayılan) ve AmbientLight yani ortamsal (belli bir merkezi ve yönü olmadan ortamda bulunan) biçimlerde olabilir. Ayrıca bir ışığın sınırları belirlenebilir, ışık o sınırlar dışına ulaşmaz. Işık nesneleri dal grubu (BranchGroup) nesnelerine bir şekil gibi eklenirler ve grubun bir parçası olurlar. Doğrusal ışık eklemek için

```
  private BranchGroup createGroup(){
    BranchGroup group = new BranchGroup();
    // ...
    Light directional=createDirectionalLight();
```

```
    group.addChild(directional);
    return group;
  }
  private Light createDirectionalLight(){
    Color3f color = new Color3f(1f, 1f, 1f);
    Point3d origin = new Point3d(0.0,0.0,0.0);
    BoundingSphere bounds = new BoundingSphere(origin, 100.0);
    Vector3f direction  = new Vector3f(4.0f, -7.0f, -12.0f);
    DirectionalLight light = new DirectionalLight(color, direction);
    light.setInfluencingBounds(bounds);
    return light;
  }
```

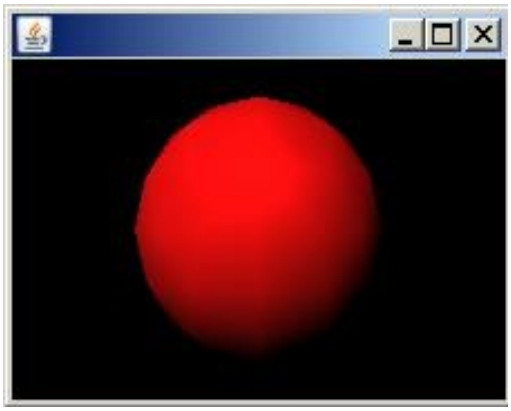
şeklinde bir kod yazılabilir. Burada bir ışığın rengi, yönü ve sınırları verilmektedir.

Ortam ışığı için oluşturmak için de

```
  private Light createAmbientLight(){
    Color3f color = new Color3f(.5f,.5f,.5f);
    AmbientLight light = new AmbientLight(color);
    Point3d origin = new Point3d(0.0,0.0,0.0);
    BoundingSphere bounds = new BoundingSphere(origin, 100.0);
    light.setInfluencingBounds(bounds);
    return light;
  }
```

şeklinde bir kod yazılır.

Yukarıdaki kodda küre şekline kırmızı ortam ışığı eklenmiştir ve aşağıdaki şekil ortaya çıkmıştır.



Şekil 2.4: 3D kütüphanesinde ışığın kullanımına örnek

2.3.8 Appearance (Görünüm), Material (Malzeme) & Texture (Doku)

Üç boyutlu nesnelerin fiziksel boyutları dışında Appearance sınıfıya belirtilen görünümleri olur. Görünümün en önemli bileşenleri Material sınıfıyla belirtilen malzeme, Texture sınıfıyla belirtilen doku özellikleridir. Material sınıfı maddenin rengini ve yaydığı ışığı tanımlar. Texture ise bir resim olabildiği gibi belli bir formüle göre görünüşü belirleyen bir yapı da içerebilir. Bir malzeme şu şekilde yüklenebilir:

```
private Material createMaterial() {
  Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
  Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
  Color3f red = new Color3f(0.7f, .15f, .15f);
  Material material= new Material(white, red, white, red, 1.0f);
  return material;
}
```

Burada malzemeye *ambien*t, *emissiv*, *diffuse* gibi malzemesel renk özellikleri ve parlaklık tanımı yapılmaktadır.

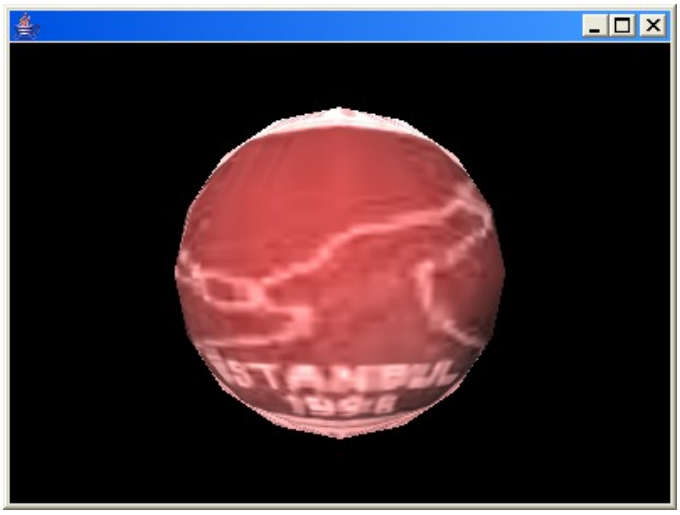
Doku yüklemek için de

```
private Texture createTexture(){
  String file=" F:\\tez\\thesis_v2\\halicArma.JPG";
```

```
TextureLoader loader=new TextureLoader(file,"LUMINANCE",new Container());
Texture texture = loader.getTexture();
texture.setBoundaryModeS(Texture.WRAP);
texture.setBoundaryModeT(Texture.WRAP);
Color4f color= new Color4f( 0.0f, 1.0f, 0.0f, 0.0f );
texture.setBoundaryColor( color);
return texture;
}
```

şeklinde bir kod yazılabilir. Burada bir resim dokuya yüklenmekte ve renk gibi parametreler ayarlanmaktadır. BoundaryMode olarak S ve T, dokunun kendi düzlemi üzerinde yatay ve dikey yönde resmin dokunun eksik kısımlarını nasıl tamamlayacağını belirtmektedir. WRAP değeri, resmi yüzey boyunca tekrar tekrar koyulması anlamına gelir.

Yukarıdaki kodun sonucunda, küre üzerinde Haliç Üniversitesi arması yerleştirilmiştir.



Şekil 2.5: 3D doku ekleme örneği

2.3.9 Transformation (Dönüşüm)

Bir veya daha fazla şeklin belli bir konum değişikliği yapması dönüşüm (transform) işlemidir. Bir dönüşüm birden fazla dönüşümü, örneğin bir kaç öteleme

(translate) ve dönme (rotate) işlemini içerebilir. Üç boyutta öteleme veya dönme gibi dönüşüm işlemleri Transform3D sınıfıyla gösterilir. Bir veya daha fazla dönüşüm işlemini birleştirmek için TransformGroup sınıfı kullanılır. Dönüşüm uygulanacak şekiller transform grubuna eklenir. Bir şekle çeşitli dönüşümlerin uygulanması çeşitli dönüşümler içeren bir dönüşüm grubuna bir şeklin eklenmesiyle olur. Bir dönüşüm grubuna birden fazla şekil eklenerek çok sayıda şeklin aynı dönüşümlere tabi olması sağlanabilir.

Basit bir ötemele dönüşümü

```
 private TransformGroup createTransform(float x,float y,float z){
   TransformGroup group = new TransformGroup();
   Transform3D transform = new Transform3D();
   Vector3f vector = new Vector3f(x, y, z);
   transform.setTranslation(vector);
   group.setTransform(transform);
   return group;
 }
```

şeklinde yapılabilir. Burada verilen x, y ve z koordinatları kadar öteleme yapan bir dönüşüm grubu oluşturulmaktadır.

Bir dal grubuna bir şekli dönüştürüp eklemek için

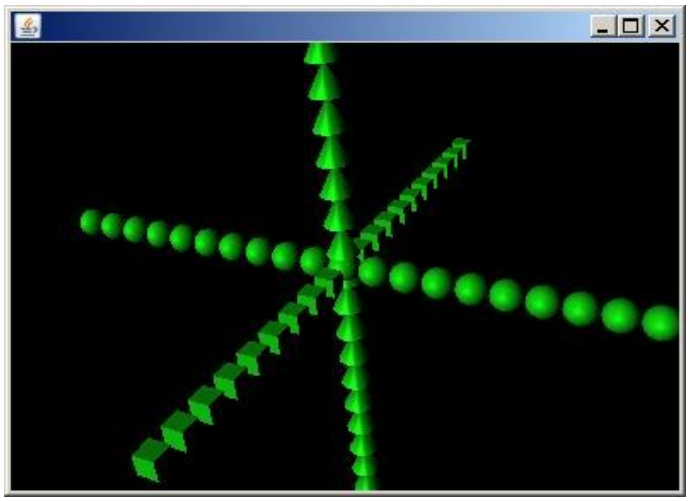
```
BranchGroup group = new BranchGroup();
Sphere sphere = new Sphere(0.05f);
TransformGroup transform=createTransform(3.0f,4.0f,5.0f);
transform.addChild(sphere);
group.addChild(transform);
```

şeklinde bir kod yazılır. İlgili kodda bir küreye (3,4,5) miktarında dönüşüm uygulanır. Sonra dönüşüm grubu dal grubuna eklenir. Dönüşüm grubu aslında dal grubunun koordinat sistemine göre belli bir dönüşüm uygulanmış yeni bir koordinat sistemidir. Her dönüşüm, eklendiği grubun koordinat sistemine göre tanımlıdır.

Üç boyutlu uzayda üç boyutun eksenleri boyunca küre (Sphere), koni (Cone) ve kutu (Box) şekillerini dizmek için

```
private BranchGroup createGroup(){
   BranchGroup group = new BranchGroup();
   for (float x = -1.0f; x <= 1.0f; x = x + 0.1f){
     Sphere sphere = new Sphere(0.05f);
     TransformGroup transform=createTransform(x,.0f,.0f);
     transform.addChild(sphere);
     group.addChild(transform);
   }
   for (float y = -1.0f; y <= 1.0f; y = y + 0.1f){
     Cone cone = new Cone(0.05f, 0.1f);
     TransformGroup  transform=createTransform(.0f, y, .0f);
     transform.addChild(cone);
     group.addChild(transform);
   }
   for (float z = -1.0f; z <= 1.0f; z = z+ 0.1f){
     Box box = new Box(0.03f,0.03f,0.03f,null);
     TransformGroup transform=createTransform(.0f, .0f, z);
     transform.addChild(box);
     group.addChild(transform);
   }
   return group;
 }
```

kodu yazılabilir ve neticesinde aşağıdaki görüntü oluşur.



Şekil 2.6: 3D dönüşüm örneği

2.3.10 Animation (Canlandırma) & Interaction (Etkileşim)

Üç boyutlu şekillerle canlandırma (animasyon) yapmak mümkündür. Animasyon için Java3D'ye özel bir kod yazmak gerekli değildir, belli bir zamanda şekilleri veya konumlarını değiştirmek yeterlidir. Hareket içeren animasyonlar, bir dal grubuna belli bir dönüşüm uygulamaktan ibarettir. Örnek olarak bir aşağı bir yukarı giden bir küre yapılabilir. Bunun için belli bir zamanda yapacağı hareketi tanımlayan bir yöntem yazmak gerekir:

```
public void move() {
  height += .1 * sign;
  if (Math.abs(height *2)  >= 1 ){
    sign = -1.0f * sign;
  }
  Vector3d vector=new Vector3d(1.0, .8, 1.0);
  if (height<-0.4f) {
    vector=new Vector3d(1.0, .8, 1.0);
  }else {
    vector=new Vector3d(1.0, 1.0, 1.0);
  }
  transform.setScale(vector);
```

```
  Vector3f translation=new Vector3f(x,height,0.0f);
  transform.setTranslation(translation);
  group.setTransform(transform);
 }
```

Burada group adındaki dal grubuna bir dönüşüm uygulanmaktadır. Dönüşüm için yükseklik (height) sürekli değiştirilmektedir. Değişimin yönü sign parametresiyle belirlenmekte, belli bir sınıra ulaştığında sign değişkeni ters işaret almakta ve kürenin hareket yön ters yönde değişmektedir.

Hareket ettiren yöntemin bir canlandırma oluşturması için bu yöntemin zaman içine tekrar tekrar çağrılmasını bir akış (thread) veya zamanlayıcı (timer) aracılığıyla sağlamak gerekir. Bunun için javax.swing.Timer sınıfı kullanılabilir :

```
 ActionListener listener=new ActionListener(){
  public void actionPerformed(ActionEvent e ) {
   move();
  }
 };
 timer = new Timer(100,listener);
```

Burada yapılan 100 milisaniye arayla hareket ettiren yöntemin çağrılmasını sağlamaktır.

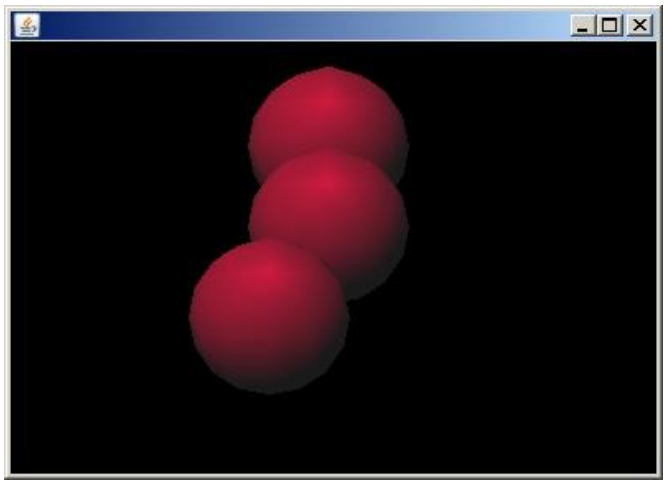
Canlandırma şekilleri zamana bağlı olarak konum veya şekil değişikliği demektir. Ancak oyun gibi konularda değişimi zamanın yerine kullanıcının belirlemesi, yani etkileşim (interaction) gerekebilir. Başka bir deyişle kullanıcı fare veya klavye hareketiyle şekillerin konumu veya biçimi üzerinde değişiklik yapabilir. Örnek olarak aşağı yukarı hareket eden kürenin kullanıcı tarafından sağa ve sola doğru hareket etmesi sağlanabilir. Bunun için klavyeyi KeyListener olarak dinlemek yeterlidir. Tuşa basıldığında

```
 public void keyPressed(KeyEvent e) {
  if (e.getKeyCode()==KeyEvent.VK_RIGHT) {
   x+=.1f;
  }else if (e.getKeyCode()==KeyEvent.VK_LEFT){
```

```
    x-=.1f;
  }
 }
```

şeklinde sola veya sağa tuşuna basılmasına göre x koordinatı değiştirilir. Zamanlayıcının bir sonraki zaman diliminde hareket ettiren *move()* yöntemini çağrılmasıyla şekil yeni konumunda dönüşüm uygulanarak gösterilecektir.

Bahsi geçen kodların neticesinde aşağıdaki şekilde bir animasyon yaratılabilir.



Şekil 2.7: 3D animasyon örneği

2.3.11 Sonuç

Java 3D ile yapılabilecekler yukarıda anlatılanlardan ibaret değildir. 3D grafiklerinde olan standart tüm işlemler, şekil değişikliği (Morphing), üç boyutlu ses (Audio 3D), başka üç boyutlu çizim formatlarının tanınması (örneğin Lightwave) gibi özellikler Java3D'de bulunmaktadır. Üstelik java3D kütüphanesi her gün gelişimini sürdüren bir kütüphanedir. Bu açıdan ilerleyen zamanlarda çok daha verimli hale gelececeği düşünülebilir. Ayrıca Java3D'nin OpenGL/DirectX'le doğrudan çalışan kütüphanelere göre nesneye yönelik ve daha modüler olduğu söylenebilir.

2.4 Projede Kullanılan 3D Algoritması

Proje mekanik fizik hareketlerinin 3 boyutlu simülasyonunu içerdiği için yukarıda bahsedilen metotlardan Animation (canlandırma) metodu uygulanmıştır.

Objeler universe içinde birbirlerine bağlanıp transform edilmişlerdir. x,y,z koordinatlarındaki konumları da, zamana bağımlı olarak, her bir simülasyonun konusu olan fizik hareketinin formülleri ile hesaplanmıştır. Bu hesaplamalar neticesinde 3 boyutlu düzlemde, zamana bağımlı olarak cismin hareketi, dolayısı ile de simülasyonu sağlanmıştır.

3 DENEYSEL & SİMULASYON

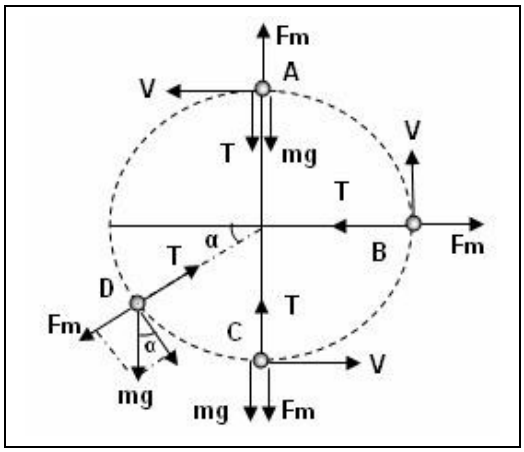
3.1 Fizik Hareketleri

Aşağıda proje için kullanılan mekanik fizik hareketleri hakkında özet bilgiler sunulmuştur.

3.1.1 Düzgün Doğrusal Hareket

Düzgün dairesel hareket yörüngesi çember olan bir harekettir. Cisim sürekli bir çember çizer.

Düzgün dairesel harekette cismin dönme hızının büyüklüğü sabittir. Yani cisim hızlanmadan veya yavaşlamadan döner.



Şekil 3.1: Düzgün dairesel hareket

Yarıçap (r) : Dairesel yörüngenin yarıçapıdır. Birimi SI birim sistemlerinde metre (m)'dir.

Periyot (T): Dairesel hareket yapan cismin bir tam tur atması için geçen süredir. Birimi SI birim sistemlerinde saniye (s) dir.

Frekans (f): Dairesel hareket yapan cismin birim zamanda attığı tur sayısıdır. Birim zaman olarak saniye alınırsa, frekans birimi SI birim sistemlerinde 1/s veya Hertz (Hz) olur.

$$T = \frac{1}{f}$$

Çizgisel hız (V): Dairesel hareket yapan cismin birim zamanda aldığı yoldur. Birim zaman olarak saniye yol olarak metre alınırsa, hız birimi SI birim sistemlerinde m/s olur. Yörüngeye her zaman teğettir.
Cisim bir tam tur döndüğünde 1 periyotluk zaman geçer ve bu sürede cisim dairenin çevresi kadar yol alır. Birim zamandaki yolu bulmak için çevreyi periyoda bölmemiz gerekir.

$$V = \frac{2\pi r}{T}$$

Açısal hız [veya Açısal frekans](w):Dairesel hareket yapan cismin birim zamanda taradığı açıdır. Birim zaman olarak saniye açı olarak radyan alınırsa, açısal hız birimi SI birim sistemlerinde rad/s olur.
Cisim bir tam tur döndüğünde 1 periyotluk zaman geçer ve bu sürede cisim 1 tam açı kadar açı tarar. Birim zamanda taranan açıyı bulmak için tam açıyı periyoda bölmemiz gerekir.

$$W = \frac{2\pi}{T}$$

Merkezcil ivme (a): Dairesel hareket yapan cismin birim zamandaki hız vektörü değişimidir. Yönü sürekli merkeze doğrudur. Birim zaman olarak saniye yol olarak metre alınırsa, merkezcil ivme birimi SI birim sistemlerinde m/s2 olur.

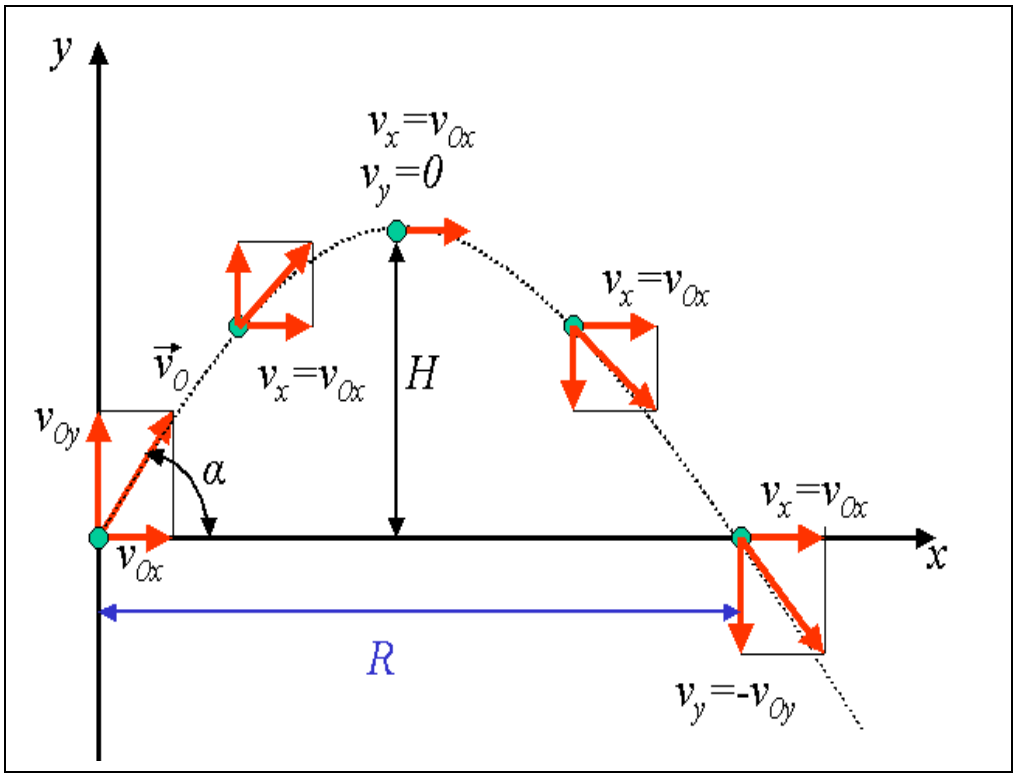
$$a = \frac{V^2}{r}$$

Merkezcil kuvvet (F): Dairesel hareket yapan cismi sürekli merkeze doğru çeken kuvvettir. Örneğin Dünya Güneş çevresinde dönerken merkezcil kuvvet Güneş'in çekim kuvvetidir. Araba virajı alırken merkezcil kuvvet sürtünme kuvvetidir. Birimi SI birim sistemlerinde Newton (N) dur. Kütle ile ivmenin çarpımı kuvveti vereceğinden, merkezcil kuvvet aşağıdaki formülden bulunur.

$$F = m\frac{V^2}{r}$$

3.1.2 Eğik Atış

Bir beysbol topunun (veya bu amaçla havaya fırlatılan herhangi bir cismin) hareketini izlemiş olan kimse bir eğik atış hareketi gözlemlemiştir. Rastgele yönlü bir ilk hızla atılan top, bir eğri yol boyunca hareket eder. Yerçekimi ivmesi hareket süresince sabit ve aşağıya doğru farzedilip; hava direncinin etkisi de ihmal edilirse eğik atışın yörüngesinin parabol olduğu gözlemlenir.



Şekil 3.2: Eğik atış hareketi

Referans sistemimizi, y doğrultusu düşey ve yukarı yön pozitif olacak şekilde seçersek, (hava direnci ihmal edildiğinden)

$$ay = -g$$

ve

$$ax = 0$$

olur.

Vx bileşeni hareket boyunca sabit kalırken, Vy bileşeni ise, cisim yukarı yönde giderken azalır, aşağı yönde giderken artar. ay bileşeni de negatif olduğundan Vy devamlı azalır.

Hareketin hız ve konum bileşenleri aşağıdaki şekildedir.

Yatay hız bileşeni: $Vx = V_0.Cos\alpha$

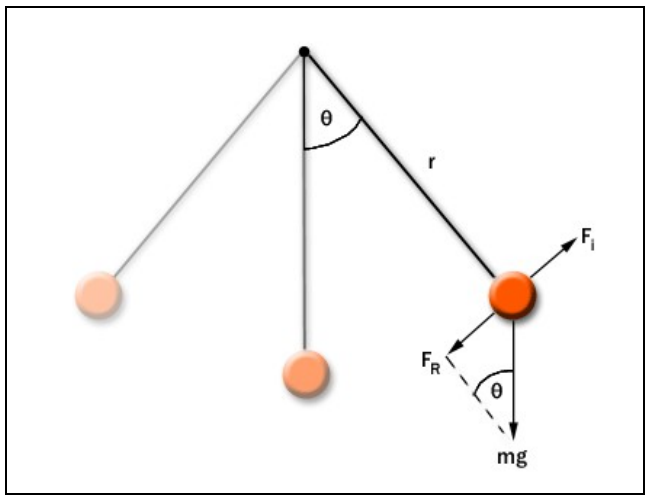
Düşey hız bileşeni: $Vy = V_0.Sin\alpha - gt$

Yatay konum bileşeni: $x = (V_0 Cos\alpha).t$

Düşey konum bileşeni: $y = (V_0 Sin\varepsilon).t - \frac{1}{2}gt^2$

3.1.3 Basit Harmonik Hareket (Sarkaç)

Bir ucundan tavana asılmış ipin diğer ucuna bir cisim bağladığımızda basit sarkaç elde etmiş oluruz. Sarkaca bir hız kazandırdığımızda sarkaç basit harmonik hareket yapmaya başlar. Eğer sürtünme yoksa, cisim sonsuza dek salınım hareketini sürdürür.



Şekil 3.3: Basit harmonik hareket

Sarkacın hareketinin bileşenleri,

Periyot: $$T = 2\pi \sqrt{\frac{r}{g}}$$

Frekans: $$f = \frac{1}{2\pi} \sqrt{\frac{g}{r}}$$

maksimum yükseklik: $$h_{\max} = \frac{V_0^2 \sin^2 \theta}{2g}$$

Menzil: $$X = \frac{V_0^2 \sin^2 \theta}{g}$$

denklemleriyle bulunur.

3.1.4 Enerjinin Korunumu, Kinetik ve Potansiyel Enerji

Enerji; iktisatçılar için yakıt anlamına gelen, fen adamları için ise varoluşun temel şekillerinden biri olup madde ile eşdeğer olan ve maddeye dönüştürülebilen bir kavramdır. Sistemdeki toplam enerji miktarı sabit olduğu deneylerle ispatlanmıştır.

Projenin konusu olan mekanik enerjiyi ele alındığında 3 tip enerji ile karşılaşılır. Bunlar kinetik enerji, potansiyel enerji ve ısı enerjisidir. Sistemdeki enerji bu enerji tipleri arasında devamlı dönüşüm halindedir ve toplam enerji korunmaktadır. Sistemdeki enerjiyi

$$E_s = E_K + E_P + E_{ISI}$$

olarak yazabiliriz.
Kinetik ve Potansiyel enerjin formülleri aşağıdaki şekildedir.

$$E_K = \frac{1}{2} m V^2$$

$$E_P = mgh$$

3.2 Tasarım ve Geliştirme

Proje geliştirim safhasında, bir 3D projesi olmasına rağmen, modern Java tasarım kalıplarına sadık kalınmaya özen gösterilmiştir. Ekranda görünen görüntü ile görüntüyü hazırlayan sınıflar birbirlerinden ayrı modellenerek izolasyon sağlanmış ve MVC (model view controller) tasarım kalıbı uygulanmıştır[6]. Projenin paket/sınıf hiyerarşisi aşağıda belirtilmiştir.

3.2.1 Paket/sınıf Hiyerarşisi

Proje,

com.sim.launch
com.sim.simulators
com.sim.view
com.sim.view.common

şeklinde 4 ana paketten oluşmaktadır. launch paketi uygulamanın asıl çalışan sınıfını (main class) barındırır. Bu sınıftan bir obje yaratıldığında program çalışmaya başlar ve ekrana uygulamanın ana ekranı gelir.

Simulators paketinde, 5 hareket için 5 ayrı simulator sınıfı vardır. Bu sınıflar, ilgili hareketin görüntü kısmından bağımsız olarak, mekanik fizik formüllerine göre sonuç üretirler. Ancak bir simulator ekranı tarafından çağırdıklarında anlam kazanırlar.

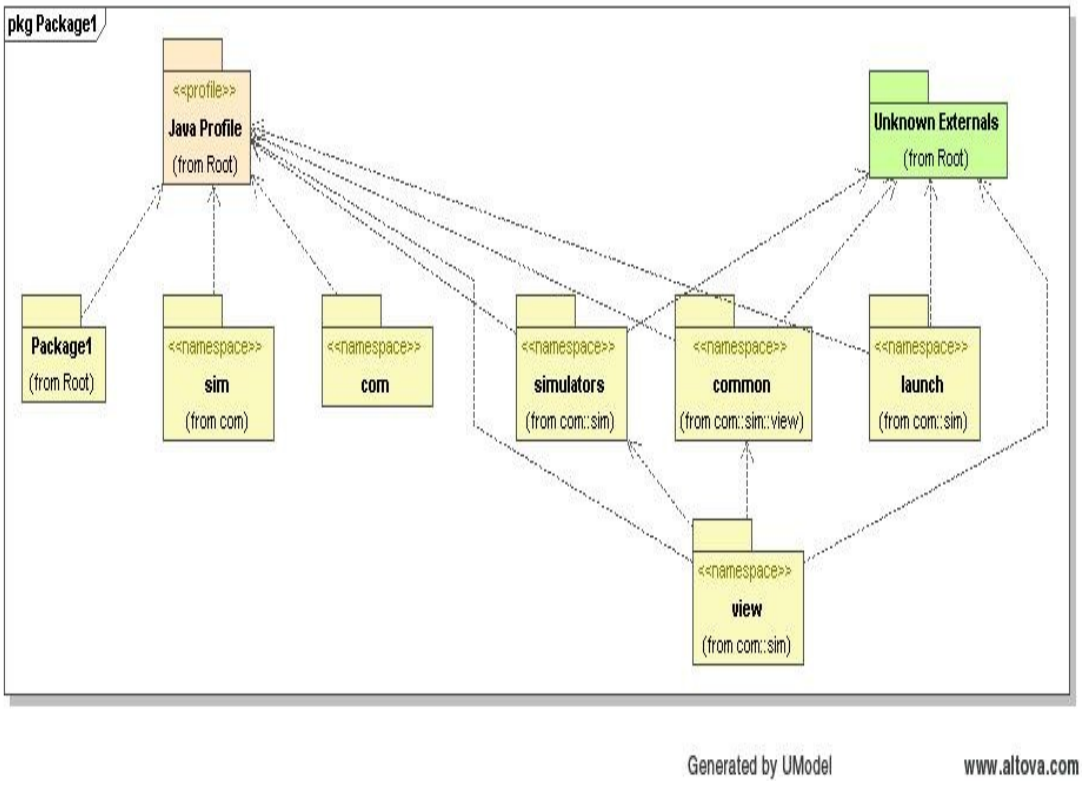
View paketinde 5 adet, her bir fizik hareketine uygun GUI (ekran) sınıfı vardır. Ana penceredeki menüden hangi hareket seçildiyse, bu paket altındaki ilgili sınıfın bir objesi yaratılır ve ekrana ilgili hareketin formu gelir. Kullanıcı herhangi bir işlem hesaplaması istediğinde, bu paket altındaki sınıflar, simulator paketi altındaki uygun sınıfları çağırırlar.

Common paketi, uygulama için ortak görüntü sınıflarını içerir. 3 boyutlu hareket eden top, hareket hakkında bilgi veren bilgi penceresi ekranı sınıfları bu paket altındadır. Paketin içerdiği en önemli dosya ise, bir interface olan

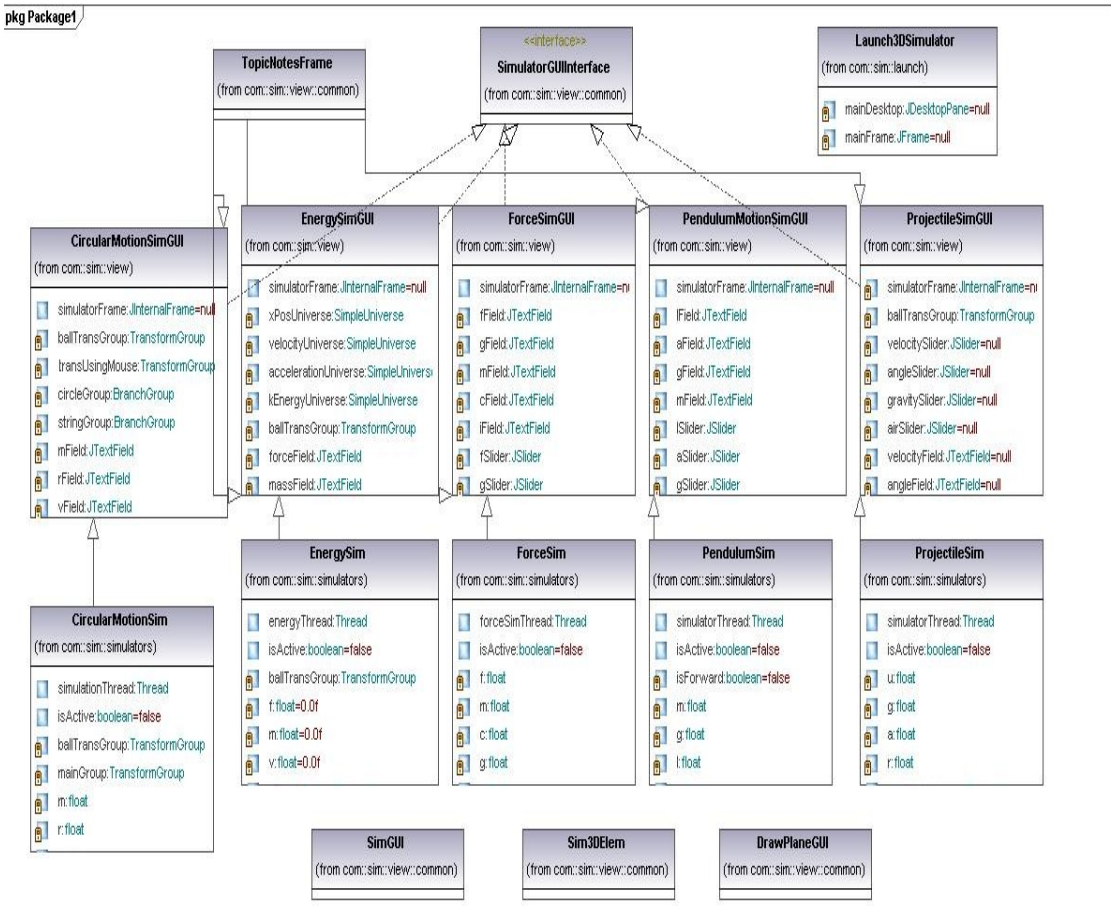
[6] : Fraser Speirs, Model-View-Controller in Web 2.0

SimulatorGUIInterface interface (arayüz)'dir. Bütün hareket ekran sınıfları bu interface'i çağırmakta, nasıl davranacaklarına interface aracılığı ile karar vermektedirler.

Aşağıda uygulamanın paket ve sınıf yapılarının UML (unified modelling language) modeli görülebilir.



Şekil 3.4: kaynak kodunun paket yapısı



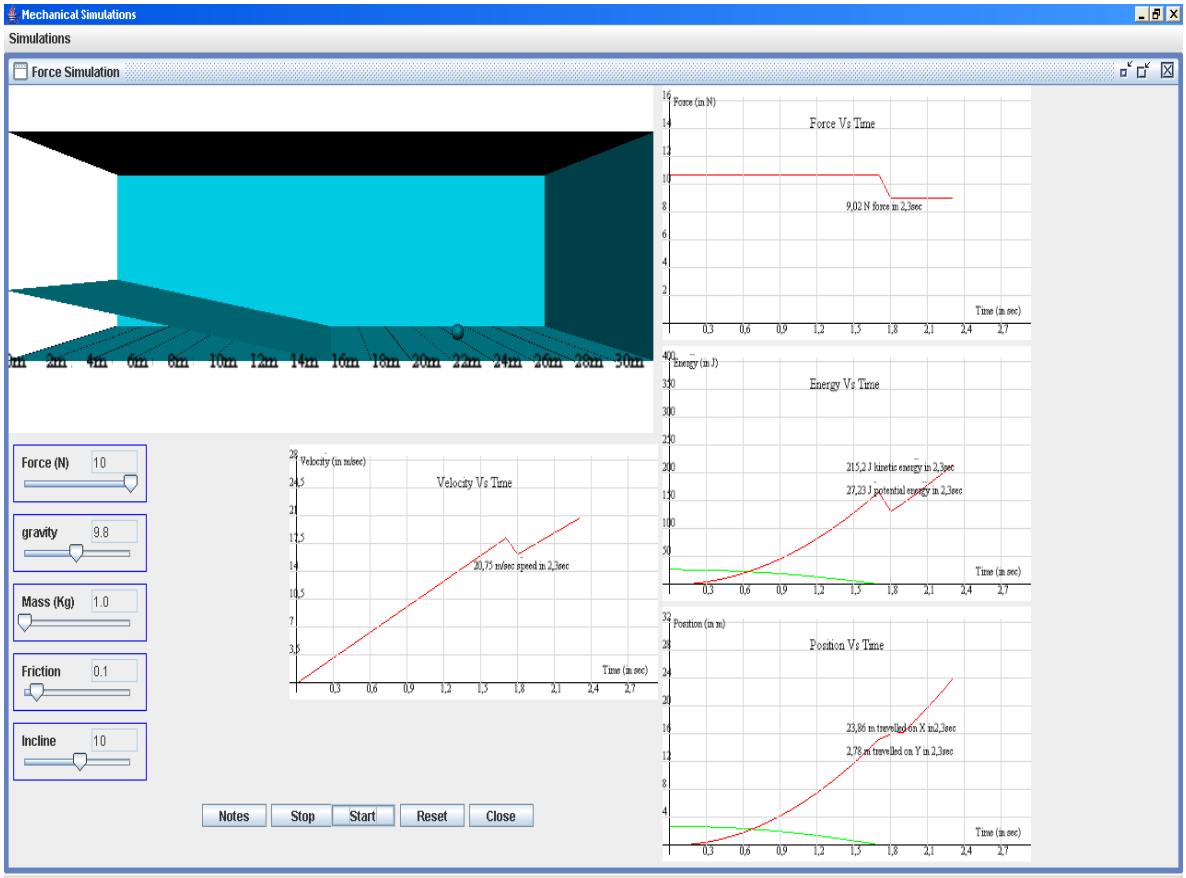
Şekil 3.5: kaynak kodunun sınıf yapısı

4 SİMULASYON SONUÇLARI

Proje alışmaları sonucunda 5 adet mekanik fizik hareketinin 3 boyutlu ortamda simülasyonu başarıyla sonuçlanmıştır. Kullanıcı tarafından dışarıdan interaktif şekilde girilen değerler, her hareket için kendi formüllerinde işlem görmüş ve 3boyutlu ortamda hareketi için x,y,z düzlemlerinde yer bilgileri hesaplanmıştır. Sürtünme kuvveti de simülasyonlara eklenmiştir. Bütün simülasyonlarda yer değiştirme, potansiyel enerji, kinetik enerji ve hız grafiklere gerçek zamanlı olarak gösterilmiştir.

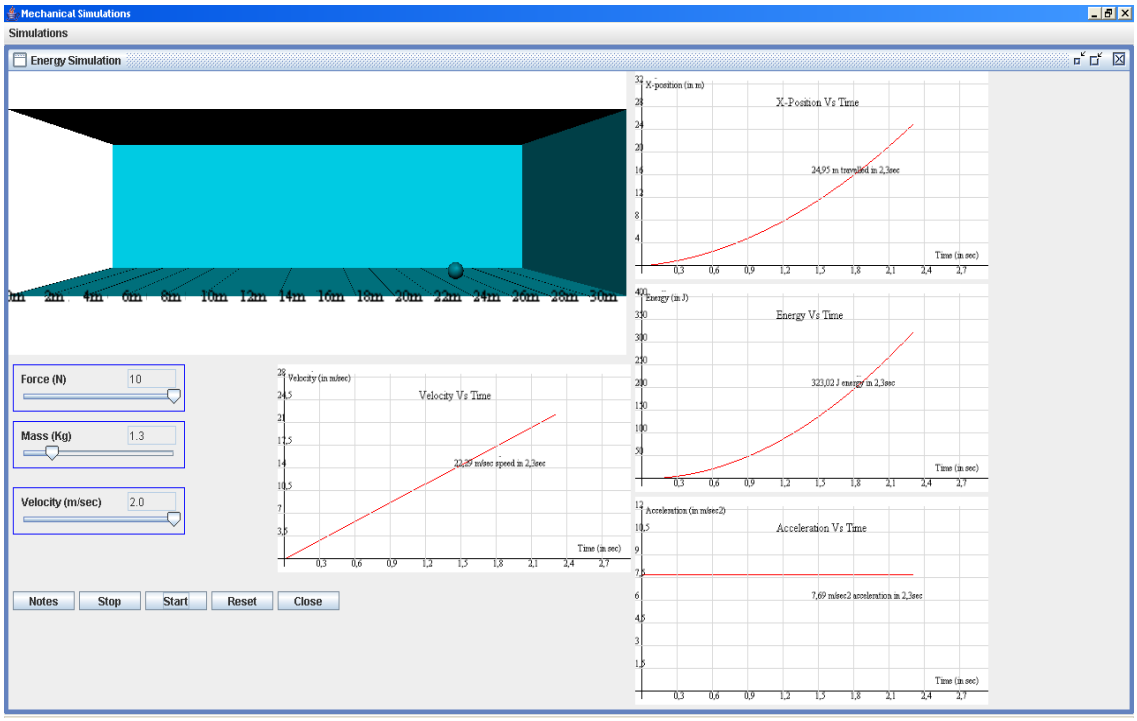
Aşağıda her bir hareketin simülasyon görüntüsü yer almaktadır.

4.1 Kuvvet Simülasyonu



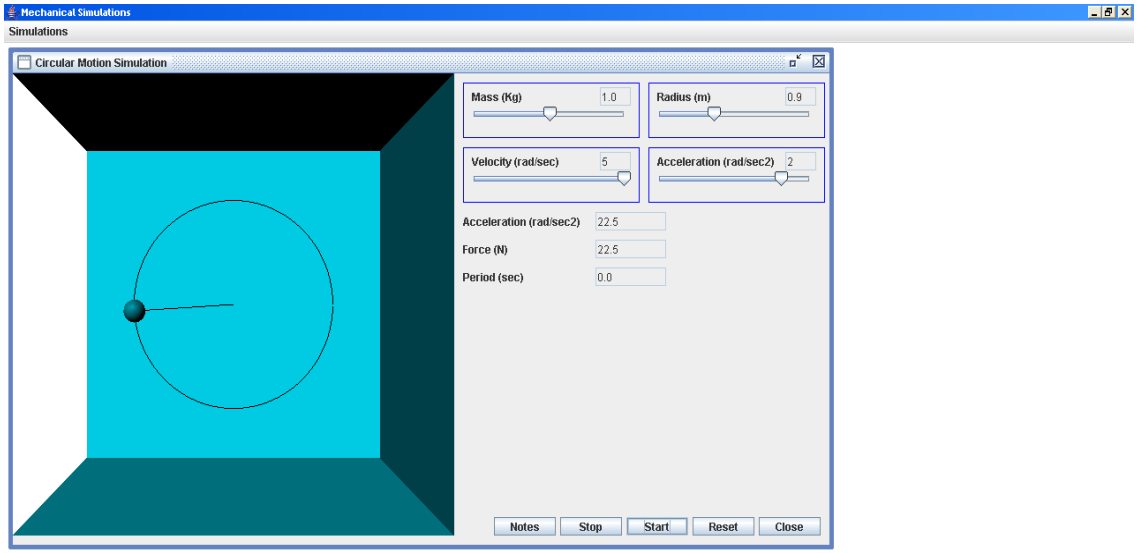
Şekil 4.1: Cisme uygulanan kuvvetin etkisinin simülasyonu

4.2 Enerji Simülasyonu



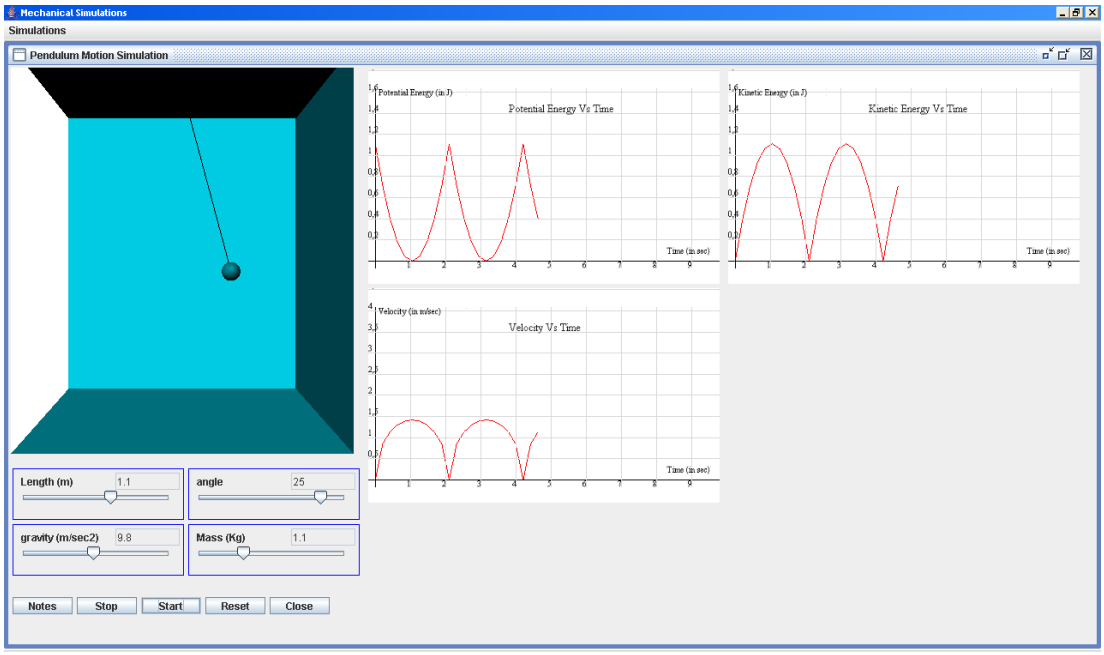
Şekil 4.2: enerjinin korunumunun simülasyonu

4.3 Düzgün Doğrusal Hareket Simülasyonu



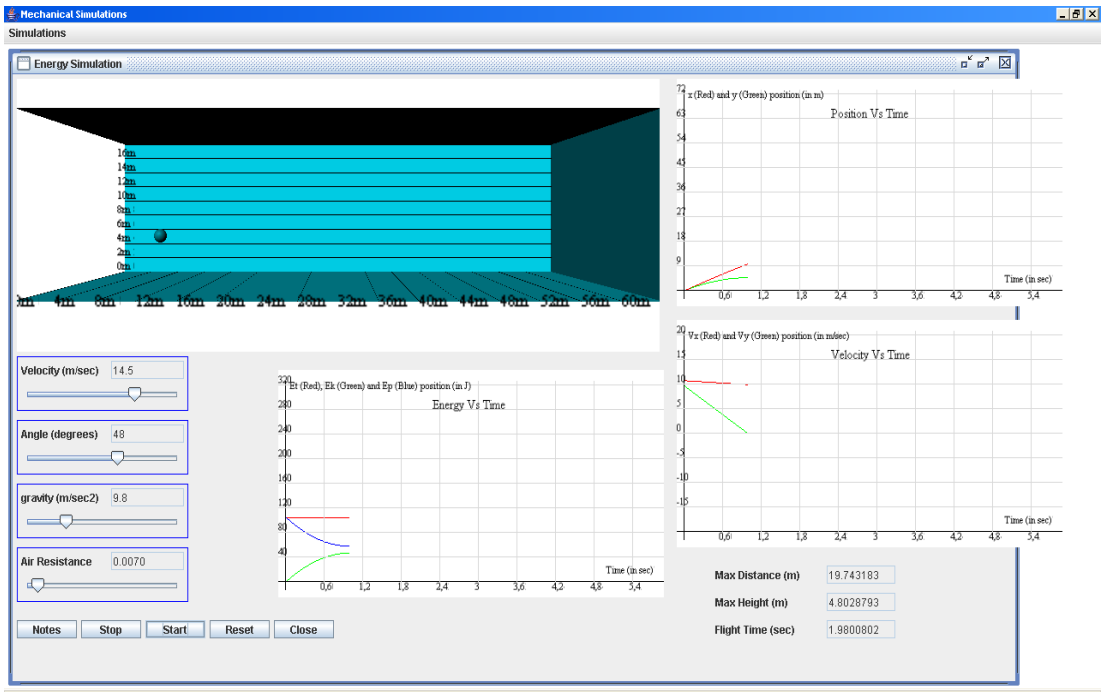
Şekil 4.3: düzgün doğrusal hareket simülasyonu

4.4 Basit Harmonik (Sarkaç) Hareket Simülasyonu



Şekil 4.4: basit harmonik hareket simülasyonu

4.5 Eğik Atış Simülasyonu



Şekil 4.5: eğik atış simülasyonu

5 TARTIŞMA & SONUÇ

İlgili çalışma neticesinde, java 3d kütüphanesinin 3 boyutlu simülasyonlar için uygun bir kütüphane olduğu gösterilmiştir. Modüler yapısı, kolay programlanabilir oluşu, karmaşık matemaik işlemlerinden programcıyı izole etmesi, bellek yönetimini üstlenmesi, yeterli performansı ile birçok uygulamada birinci seçenek olarak öne çıkabileceği anlaşılmıştır.

Ayrıca proje ile hangi tür simülasyonlar yapılabileceğine ışık tutulmuş, yeni fikirlerin üretilmesi, üretilen fikirlerin sistemlerinin modellenmesi için uygun altyapı alternatifi sunulmuştur.

Java dilinin web teknolojilerine olan yakınlığı sebebiyle, web üzerinden 3 boyutlu uygulamaların da rahatlıkla geliştirilebileceği gösterilmiştir. Internet üzerinden yapılan alışverişlerde, kullanıcıya sanal gerçeklik sağlanabileceği anlaşılmıştır.

Ayrıca seçilen konu gereği, fizik bilimi öğrencilerine anlaşılır simülasyonlar yapılabileceği görülüp, eğitim alanında ve her tür pozitif bilimde rahatlıkla uygulanabileceği fark edilmiştir.

Yukarıda sayılan maddeler ışığında, ilgili proje fizik bilimi gibi pozitif ve dünyada geçerliliği olan bir sistem ile 3 boyutlu sanal gerçeklik dünyasını birleştirmiş; eğitim, bilim ve sosyal hayat alanına katkılar sağlayabileceğini göstermiştir.

5.1 Projede Yapılmayanlar / Neler Yapılabilir?

Proje çerçevesinde java3d dışında başka bir dil ile örnek yapılmamıştır. 3D kütüphanelerinin performans kıyaslanması açısından başka dillerle (OpenGL, DirectX, vrml) örneklerin bulunmasında fayda olacaktır. Bellek kullanımları, işlemci kullanım oranları, render kaliteleri açısında mukayeseli sonuçlar birçok soruya cevap vermesi açısından yol gösterici olabilir.

Mekanik fizik hareketleri dışında başka sistem örnekleri de yapılması, 3D simülasyon konusunda ufuk açıcı ve fikir verici olacaktır. Özellikle reel dünyaya ait simülasyon örnekleri, 3D simülasyonların öneminin anlaşılmasına ve daha çok kullanılmasının teşvikine vesile olacaktır.

KAYNAKLAR

[1] Harvey M.Deitel, Paul J.Deitel, Java How To Progam (5th Ed.), Deitel & Deitel

[2] Daniel Selman, Java 3D Programming, Manning, 2004

[3] Cay S.Horstmann, Gary Cornell, Core Java Volume 1 Fundementals, Prentice Hall

[4] Sowizral Henry, Rushforth Kevin, The Java 3D API Specification (2nd Ed.), Addison Wesley

[5] Gene Davis, Learning Java Binding for OpenGL, AuthorHouse

[6] Oliver Faulhaber, A Scene-graph Editor for Java 3D, 2005

[7] Fraser Speirs, Model-View-Controller in Web 2.0

[8] Fikri Öztürk, Levent Özbek, Matematiksel Modelleme ve Simülasyon, Gazi Yayınevi

[9] Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL Programming Guide (5th Ed.), Addison-Wesley

[10] Wayne E. Carlson, An Algorithm and Data Structure for 3D Object Synthesis Using Surface Patch Intersections

[11] Jon Barrilleaux, 3D User Interfaces with Java3D, Manning

[12] Richard S.Wright Jr, Michael Sweet, OpenGL SuperBible (2nd Ed.), Waite Group Press

[13] Daivid Luebke, Jonathan D.Cohen, Level of 3D Graphics, Morgan Kaufmann

[14] Julio Sanchez, Maria Canton, DirectX 3D Graphics Programming, Hungry Minds

[15] Peter Shirley, Fundemantals of Computer Graphics (2nd Ed.), A K Peters, Ltd

[16] Karl F.Kuhn, Basic Physics (2nd Ed.), Wiley

[17] Frank Klawonn, Introduction to Computer Graphics: Using Java 2D and 3D, Springer

EK 1 (Paket Yapısı ve Örnek Kaynak Kodu)

```
class PendulumMotionSimGUI
    JInternalFrame loadMainGUI(JFrame mainFrame)
    loadGraphs()
    loadInputOutputGUI()
    loadSimulatorGUI()
    startSimulation()
    actionPerformed(ActionEvent arg0)
    stateChanged(ChangeEvent arg0)
    changeBallLocation()
    clearTrace()

class Launch3DSimulator
    loadMainGUI
    createMenuBar()
    main(String[] args)

class CircularMotionSim
    CircularMotionSim(TransformGroup ballTransGroup, TransformGroup
mainGroup, BranchGroup stringGroup, float m, float r,
                    float v, float a)
    run()
    stop()

class EnergySim
    EnergySim(TransformGroup ballTransGroup, int force, int mass, int velocity,
SimpleUniverse posUniv,
                    SimpleUniverse velUniv, SimpleUniverse accUniv,
SimpleUniverse kEUniv, BranchGroup xPosGroup[], BranchGroup velGroup[],
                    BranchGroup accGroup[], BranchGroup kEGroup[])
    run()
    stop()
    class ForceSim
    run()
    stop()


class ProjectileSimGUI
    loadMainGUI
    loadSimulatorGUI()
    startSimulation()
    clearTrace()
    loadGraphs()
    loadInputOutputGUI()
    actionPerformed(ActionEvent arg0)
    stateChanged(ChangeEvent arg0)
```

```
interface SimulatorGUIInterface
      loadMainGUI(JFrame mainFrame);
      void loadInputOutputGUI();
      public void loadSimulatorGUI();
      public void loadGraphs();
      public void startSimulation();


class SimGUI
      JInternalFrame cIFrame(String frameTitle, Rectangle frameBounds)


class DrawPlaneGUI
      DrawPlaneGUI(Point3f A, Point3f B, Point3f C, Point3f D)
      Geometry createGeometry(Point3f A, Point3f B, Point3f C, Point3f D)
      Appearance createAppearance()
```

SimulatorGUIInterface.java

```
package com.sim.view.common;

import javax.swing.JFrame;
import javax.swing.JInternalFrame;

public interface SimulatorGUIInterface
{
        //Method to load simulator GUI
        public JInternalFrame loadMainGUI(JFrame mainFrame);

        //Method will load the input and output GUI
        public void loadInputOutputGUI();

        //Method will load the simulation GUI screen
        public void loadSimulatorGUI();

        //Method will load the graphs for the screen
        public void loadGraphs();

        //Method will start the actual simulations
        public void startSimulation();
}
```

ProjectileSim.java

```
package com.sim.simulators;

import javax.media.j3d.BranchGroup;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import javax.vecmath.Color3f;
import javax.vecmath.Point3f;
import javax.vecmath.Vector3f;

import com.sim.view.common.Sim3DElem;
import com.sun.j3d.utils.universe.SimpleUniverse;

public class ProjectileSim implements Runnable{

	public Thread simulatorThread;
	public boolean isActive = false;

	private float u, g, a, r;
	private SimpleUniverse positionUniverse, velocityUniverse, energyUniverse;
	private BranchGroup positionGroup[], velocityGroup[], energyGroup[];
	private TransformGroup ballTransGroup;
	public float mD = 0.0f, mH = 0.0f, fT = 0.0f;

	public ProjectileSim(TransformGroup ballTransGroup, float u, float g, float a, float c, SimpleUniverse posUniv,
			SimpleUniverse velUniv, SimpleUniverse eneUniv, BranchGroup posGrp[], BranchGroup velGrp[], BranchGroup eneGrp[]){
		simulatorThread = new Thread(this);
		this.u = u;
		this.g = g;
		this.a = a;
		this.r = c;

		float Vxo = (float)(u*Math.sin(Math.PI*(a/180.0f)));
		float Vyo = (float)(u*Math.cos(Math.PI*(a/180.0f)));
		fT = 2.0f*Vyo/g;
		mD = Vxo*fT - 0.5f*r*Vxo*Vxo*fT*fT;
		mH = Vyo*fT/2.0f - 0.125f*g*fT*fT;

		this.ballTransGroup = ballTransGroup;
		this.positionGroup = posGrp;
		this.positionUniverse = posUniv;
		this.velocityGroup = velGrp;
		this.velocityUniverse = velUniv;
		this.energyGroup = eneGrp;
		this.energyUniverse = eneUniv;
```

```
	}

	public void run(){
		float Vxo = (float)(u*Math.sin(Math.PI*(a/180.0f)));
		float Vyo = (float)(u*Math.cos(Math.PI*(a/180.0f)));

		float Et = 0.5f*u*u;
		float tPass = 0.0f;
		float dx = 0.0f, dy = 0.0f, Ek = 0.0f, Ep = 0.0f, Vx = 0.0f, Vy = 0.0f;
		int   dataPoint = 0;

		Point3f startPosX = new Point3f(-0.96f, -0.48f, 0.0f), endPosX;
		Point3f startPosY = new Point3f(-0.96f, -0.48f, 0.0f), endPosY;
		Point3f startPosEt = new Point3f(-0.96f, -0.48f, 0.0f), endPosEt;
		Point3f startPosEp = new Point3f(-0.96f, -0.48f, 0.0f), endPosEp;
		Point3f startPosEk = new Point3f(-0.96f, -0.48f, 0.0f), endPosEk;
		Point3f startPosVx = new Point3f(-0.96f, -0.48f, 0.0f), endPosVx;
		Point3f startPosVy = new Point3f(-0.96f, -0.48f, 0.0f), endPosVy;
		isActive = true;
		for(int i=0; i<21 && isActive; i++){
			Vx = Vxo - r*Vxo*Vxo*tPass;
			Vy = Vyo - g*tPass;
			dx = Vx*tPass;
			dy = Vyo*tPass - 0.5f*g*tPass*tPass;

			//Change the position of ball
			Transform3D ballNewPosition = new Transform3D();
			ballNewPosition.setTranslation(new Vector3f(-0.8f +
dx*0.025f, -0.18f + 0.025f*dy, 0.0f));
			ballTransGroup.setTransform(ballNewPosition);

			//Plot x and y position curve with time
			endPosX = new Point3f(-0.96f + tPass/3.0f, -0.48f + 0.12f*dx/
9.0f, 0.0f);
			endPosY = new Point3f(-0.96f + tPass/3.0f, -0.48f + 0.12f*dy/
9.0f, 0.0f);
			positionGroup[dataPoint] =
Sim3DElem.createGroupWithLine(startPosX, endPosX, new Color3f(1.0f, 0.0f,
0.0f));

		positionGroup[dataPoint].addChild(Sim3DElem.createLine(startPosY,
endPosY, new Color3f(0.0f, 1.0f, 0.0f)));
			positionUniverse.addBranchGraph(positionGroup[dataPoint]);
			startPosX = endPosX;
			startPosY = endPosY;

			//Plot Et, Ek and Ep curve with time
			Ep = g*dy;
			Ek = Et - Ep;
```

```
                    endPosEt = new Point3f(-0.96f + tPass/3.0f, -0.48f + 0.12f*Et/
40.0f, 0.0f);
                    endPosEp = new Point3f(-0.96f + tPass/3.0f, -0.48f +
0.12f*Ep/40.0f, 0.0f);
                    endPosEk = new Point3f(-0.96f + tPass/3.0f, -0.48f +
0.12f*Ek/40.0f, 0.0f);
                    energyGroup[dataPoint] =
Sim3DElem.createGroupWithLine(startPosEt, endPosEt, new Color3f(1.0f, 0.0f,
0.0f));

        energyGroup[dataPoint].addChild(Sim3DElem.createLine(startPosEp,
endPosEp, new Color3f(0.0f, 1.0f, 0.0f)));

        energyGroup[dataPoint].addChild(Sim3DElem.createLine(startPosEk,
endPosEk, new Color3f(0.0f, 0.0f, 1.0f)));
                    energyUniverse.addBranchGraph(energyGroup[dataPoint]);
                    startPosEt = endPosEt;
                    startPosEk = endPosEk;
                    startPosEp = endPosEp;

                    //Plot Vx and Vy curve with time
                    endPosVx = new Point3f(-0.96f + tPass/3.0f, 0.12f*Vx/5.0f,
0.0f);
                    endPosVy = new Point3f(-0.96f + tPass/3.0f, 0.12f*Vy/5.0f,
0.0f);
                    velocityGroup[dataPoint] =
Sim3DElem.createGroupWithLine(startPosVx, endPosVx, new Color3f(1.0f, 0.0f,
0.0f));

        velocityGroup[dataPoint].addChild(Sim3DElem.createLine(startPosVy,
endPosVy, new Color3f(0.0f, 1.0f, 0.0f)));
                    velocityUniverse.addBranchGraph(velocityGroup[dataPoint]);
                    startPosVx = endPosVx;
                    startPosVy = endPosVy;

                    try{
                        dataPoint++;
                        tPass = tPass + fT/20.0f;
                        Thread.sleep((long)(fT*1000.0f/20.0f));
                    }catch(Exception e){
                    }
                }

                isActive = false;
        }

        public void stop(){
                isActive = false;
        }
}
```

ProjectileSimGUI.java

```
package com.sim.view;

import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.media.j3d.BoundingSphere;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Canvas3D;
import javax.media.j3d.DirectionalLight;
import javax.media.j3d.Shape3D;
import javax.media.j3d.TransformGroup;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.JTextField;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.vecmath.Color3f;
import javax.vecmath.Point3d;
import javax.vecmath.Point3f;
import javax.vecmath.Vector3f;

import com.sim.simulators.ProjectileSim;
import com.sim.view.common.DrawPlaneGUI;
import com.sim.view.common.Sim3DElem;
import com.sim.view.common.SimGUI;
import com.sim.view.common.SimulatorGUIInterface;
import com.sim.view.common.TopicNotesFrame;
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
import com.sun.j3d.utils.universe.SimpleUniverse;
import com.sun.j3d.utils.universe.ViewingPlatform;

public class ProjectileSimGUI implements SimulatorGUIInterface, ActionListener,
ChangeListener
{

        private JInternalFrame      simulatorFrame = null;
        private TransformGroup      ballTransGroup;
        private JSlider             velocitySlider = null, angleSlider = null,
gravitySlider = null, airSlider = null;
        private JTextField          velocityField = null, angleField = null,
gravityField = null, airField = null;
        private JTextField          maxHeightField = null, timeOfFlightField =
null, horizontalDistanceField = null;
        private SimpleUniverse      positionUniverse,           energyUniverse,
velocityUniverse;
```

```
        private BranchGroup  posGrp[] = new BranchGroup[21];
        private BranchGroup  energyGrp[] = new BranchGroup[21];
        private BranchGroup  velocityGrp[] = new BranchGroup[21];
        private ProjectileSim  projectileSim;
        private JFrame                          mainFrame;

        public JInternalFrame loadMainGUI(JFrame mainFrame)
        {
                simulatorFrame = SimGUI.cIFrame("Energy Simulation", new
Rectangle(5, 5, 1180, 700));

                //Create initial value panel and display it
                loadInputOutputGUI();
                loadSimulatorGUI();

                //Create all the graphs
                loadGraphs();

                simulatorFrame.setVisible(true);
                simulatorFrame.setResizable(true);
                simulatorFrame.setMaximizable(true);
                this.mainFrame = mainFrame;
                return simulatorFrame;
        }

        public void loadSimulatorGUI()
        {
                //Create the canvas
                Canvas3D                energyCanvas                =                new
Canvas3D(SimpleUniverse.getPreferredConfiguration());
                energyCanvas.setBounds(5, 5, 750, 300);
                simulatorFrame.getContentPane().add(energyCanvas);

                //Create the translation and rotation using mouse
                TransformGroup transUsingMouse = new TransformGroup();


transUsingMouse.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);


transUsingMouse.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE)
;

                //Create the branch group, add background and other objects
                BranchGroup simulationGroup = new BranchGroup();
                simulationGroup.addChild(Sim3DElem.createWhiteBackgroud(new
Color3f(1.0f, 1.0f, 1.0f)));

                //Create the room walls
```

```
            Shape3D topWall = new DrawPlaneGUI(new Point3f(-0.8f, 0.25f,
-0.5f), new Point3f(0.8f, 0.25f, -0.5f),
                  new Point3f(0.8f, 0.25f, 0.5f), new Point3f(-0.8f, 0.25f, 0.5f));
            transUsingMouse.addChild(topWall);

            Shape3D bottomWall = new DrawPlaneGUI(new Point3f(-0.8f, -0.2f,
0.5f), new Point3f(0.8f, -0.2f, 0.5f),
                  new Point3f(0.8f, -0.2f, -0.5f), new Point3f(-0.8f, -0.2f, -0.5f));
            transUsingMouse.addChild(bottomWall);

            Shape3D centerPlane = new DrawPlaneGUI(new Point3f(-0.8f, -0.2f,
-0.5f), new Point3f(0.8f, -0.2f, -0.5f),
                  new Point3f(0.8f, 0.25f, -0.5f), new Point3f(-0.8f, 0.25f, -0.5f));
            transUsingMouse.addChild(centerPlane);

            Shape3D rightWall = new DrawPlaneGUI(new Point3f(0.8f, -0.20f,
-0.5f), new Point3f(0.8f, -0.20f, 0.5f),
                  new Point3f(0.8f, 0.25f, 0.5f), new Point3f(0.8f, 0.25f, -0.5f));
            transUsingMouse.addChild(rightWall);

            //Create markings on floor for distance
            for(int i=0; i<16; i++)
            {
                  transUsingMouse.addChild(Sim3DElem.createLine(new
Point3f(-0.8f + 0.1f*i, -0.2f, -0.5f),
                        new Point3f(-0.8f + 0.1f*i, -0.2f, 0.5f), new
Color3f(0.0f, 0.0f, 0.0f)));
                  transUsingMouse.addChild(Sim3DElem.createText(i*4 + "m",
new Vector3f(-0.8f + 0.1f*i, -0.22f, 0.5f), 10));
            }

            //Create markings on back wall for height
            for(int i=0; i<10; i++)
            {
                  transUsingMouse.addChild(Sim3DElem.createLine(new
Point3f(-0.8f, -0.2f + 0.05f*i, -0.49f),
                        new Point3f(0.8f, -0.2f + 0.05f*i, -0.49f), new
Color3f(0.0f, 0.0f, 0.0f)));
                  transUsingMouse.addChild(Sim3DElem.createText(i*2 + "m",
new Vector3f(-0.83f, -0.2f + 0.05f*i, -0.5f), 10));
            }

            //Create the object for simulation
            ballTransGroup = Sim3DElem.createBallTransformGroup(new
Vector3f(-0.8f, -0.18f, 0.0f), 0.02f);
            transUsingMouse.addChild(ballTransGroup);

            //Set the scene ambient colors
            DirectionalLight sceneLight = new DirectionalLight(new
Color3f(0.0f, 0.9f, 1.0f), new Vector3f(4.0f, -7.0f, -12.0f));
```

```
            sceneLight.setInfluencingBounds(new BoundingSphere(new
Point3d(0.0, 0.0, 0.0), 100.0));
            simulationGroup.addChild(sceneLight);

            //Create the Universe object for all the screen items
            simulationGroup.addChild(transUsingMouse);
          simulationGroup.compile();
            SimpleUniverse simulationUniverse = new
SimpleUniverse(energyCanvas);

            ViewingPlatform viewingPlatform =
simulationUniverse.getViewingPlatform();
            BranchGroup rotateBG = new BranchGroup();
            MouseRotate myMouseRotate = new MouseRotate();
            myMouseRotate.setTransformGroup(transUsingMouse);
            myMouseRotate.setSchedulingBounds(new BoundingSphere());
            rotateBG.addChild(myMouseRotate);
            viewingPlatform.addChild(rotateBG);


        simulationUniverse.getViewingPlatform().setNominalViewingTransform();
            simulationUniverse.addBranchGraph(simulationGroup);
        }

        public void startSimulation()
        {
            try{
                if(projectileSim.isActive)
                {
                    return;
                }
            }catch(Exception e){}
            clearTrace();
            projectileSim = new ProjectileSim(ballTransGroup,
(float)velocitySlider.getValue()/10.0f,
                    (float)gravitySlider.getValue()/10.0f,
angleSlider.getValue(), (float)airSlider.getValue()/1000.0f,
                    positionUniverse, velocityUniverse, energyUniverse,
posGrp, velocityGrp, energyGrp);
            projectileSim.simulatorThread.start();
        }

        /**
         * Clear all the graphs
         */
        public void clearTrace()
        {
            for(int i=0; i<25; i++)
            {
                try
```

```
                {
                        posGrp[i].detach();
                        energyGrp[i].detach();
                        velocityGrp[i].detach();
                }catch(Exception e){
                }
        }
    }

    public void loadGraphs()
    {
        //Create the canvas
        Canvas3D posCanvas = new
Canvas3D(SimpleUniverse.getPreferredConfiguration());
        posCanvas.setBounds(775, 5, 450, 250);
        simulatorFrame.getContentPane().add(posCanvas);

        //Create the Universe object for all the screen items
        BranchGroup posGroup = Sim3DElem.createGraphBase("x (Red) and
y (Green) position (in m)", "Position Vs Time", 9.0f, 0.6f);
        posGroup.compile();
        positionUniverse = new SimpleUniverse(posCanvas);

    positionUniverse.getViewingPlatform().setNominalViewingTransform();
        positionUniverse.addBranchGraph(posGroup);

        //Create the canvas
        Canvas3D velocityCanvas = new
Canvas3D(SimpleUniverse.getPreferredConfiguration());
        velocityCanvas.setBounds(775, 270, 450, 250);
        simulatorFrame.getContentPane().add(velocityCanvas);

        //Create the Universe object for all the screen items
        BranchGroup velocityGroup = Sim3DElem.createGraphBase("Vx
(Red) and Vy (Green) position (in m/sec)",
                "Velocity Vs Time", -5.0f, 0.6f);
        velocityGroup.compile();
        velocityUniverse = new SimpleUniverse(velocityCanvas);

    velocityUniverse.getViewingPlatform().setNominalViewingTransform();
        velocityUniverse.addBranchGraph(velocityGroup);

        //Create the canvas
        Canvas3D energyCanvas = new
Canvas3D(SimpleUniverse.getPreferredConfiguration());
        energyCanvas.setBounds(310, 325, 450, 250);
        simulatorFrame.getContentPane().add(energyCanvas);

        //Create the Universe object for all the screen items
```

```
                BranchGroup energyGroup = Sim3DElem.createGraphBase("Et
(Red), Ek (Green) and Ep (Blue) position (in J)",
                                "Energy Vs Time", 40.0f, 0.6f);
                energyGroup.compile();
                energyUniverse = new SimpleUniverse(energyCanvas);

        energyUniverse.getViewingPlatform().setNominalViewingTransform();
                energyUniverse.addBranchGraph(energyGroup);
        }

        public void loadInputOutputGUI() {
                JPanel speedPanel = SimGUI.cPanel(new Rectangle(5, 310, 200, 60));
                speedPanel.add(SimGUI.cLabel("Velocity (m/sec) ", new
Rectangle(5, 5, 100, 20)));
                velocityField = SimGUI.cField("10", new Rectangle(110, 5, 85, 20),
false);
                speedPanel.add(velocityField);
                velocitySlider = SimGUI.cSlider(10, 200, 100, new Rectangle(5, 30,
190, 25), this);
                speedPanel.add(velocitySlider);
                simulatorFrame.add(speedPanel);

                JPanel anglePanel = SimGUI.cPanel(new Rectangle(5, 380, 200, 60));
                anglePanel.add(SimGUI.cLabel("Angle (degrees) ", new Rectangle(5,
5, 100, 20)));
                angleField = SimGUI.cField("45", new Rectangle(110, 5, 85, 20),
false);
                anglePanel.add(angleField);
                angleSlider = SimGUI.cSlider(30, 60, 45, new Rectangle(5, 30, 190,
25), this);
                anglePanel.add(angleSlider);
                simulatorFrame.add(anglePanel);

                JPanel gravityPanel = SimGUI.cPanel(new Rectangle(5, 450, 200,
60));
                gravityPanel.add(SimGUI.cLabel("gravity (m/sec2) ", new
Rectangle(5, 5, 100, 20)));
                gravityField = SimGUI.cField("9.8", new Rectangle(110, 5, 85, 20),
false);
                gravityPanel.add(gravityField);
                gravitySlider = SimGUI.cSlider(80, 150, 98, new Rectangle(5, 30,
190, 25), this);
                gravityPanel.add(gravitySlider);
                simulatorFrame.add(gravityPanel);

                JPanel airDragPanel = SimGUI.cPanel(new Rectangle(5, 520, 200,
60));
                airDragPanel.add(SimGUI.cLabel("Air Resistance ", new
Rectangle(5, 5, 100, 20)));
                airField = SimGUI.cField("0.0", new Rectangle(110, 5, 85, 20), false);
```

```
                airDragPanel.add(airField);
                airSlider = SimGUI.cSlider(0, 100, 0, new Rectangle(5, 30, 190, 25),
this);
                airDragPanel.add(airSlider);
                simulatorFrame.add(airDragPanel);

                simulatorFrame.add(SimGUI.cLabel("Max Distance (m) ", new
Rectangle(820, 540, 120, 20)));
                horizontalDistanceField = SimGUI.cField("", new Rectangle(950,
540, 80, 20), false);
                simulatorFrame.add(horizontalDistanceField);

                simulatorFrame.add(SimGUI.cLabel("Max Height (m) ", new
Rectangle(820, 570, 120, 20)));
                maxHeightField = SimGUI.cField("", new Rectangle(950, 570, 80,
20), false);
                simulatorFrame.add(maxHeightField);

                simulatorFrame.add(SimGUI.cLabel("Flight Time (sec) ", new
Rectangle(820, 600, 120, 20)));
                timeOfFlightField = SimGUI.cField("", new Rectangle(950, 600, 80,
20), false);
                simulatorFrame.add(timeOfFlightField);

                //Create action buttons
                simulatorFrame.add(SimGUI.cButton("Notes", new Rectangle(5, 600,
70, 20), this));
                simulatorFrame.add(SimGUI.cButton("Stop", new Rectangle(80, 600,
70, 20), this));
                simulatorFrame.add(SimGUI.cButton("Start", new Rectangle(155,
600, 70, 20), this));
                simulatorFrame.add(SimGUI.cButton("Reset", new Rectangle(230,
600, 70, 20), this));
                simulatorFrame.add(SimGUI.cButton("Close", new Rectangle(305,
600, 70, 20), this));
        }

        public void actionPerformed(ActionEvent arg0) {
                if("Notes".equals(arg0.getActionCommand())){
                        TopicNotesFrame notesFrame = new
TopicNotesFrame(mainFrame, "ProjectileMotion");
                        notesFrame.setVisible(true);
                }else if("Stop".equals(arg0.getActionCommand())){
                        try{
                                projectileSim.stop();
                        }catch(Exception e){}
                }else if("Start".equals(arg0.getActionCommand())){
                        startSimulation();
                        horizontalDistanceField.setText(projectileSim.mD + "");
                        maxHeightField.setText(projectileSim.mH + "");
```

```
                timeOfFlightField.setText(projectileSim.fT + "");
            }else if("Reset".equals(arg0.getActionCommand())){
                //Set the sliders
                velocitySlider.setValue(10);
                angleSlider.setValue(45);
                gravitySlider.setValue(98);
                airSlider.setValue(0);

                //Set the text fields
                velocityField.setText(velocitySlider.getValue()/10.0f + "");
                angleField.setText(angleSlider.getValue() + "");
                gravityField.setText(gravitySlider.getValue()/10.0f + "");
                airField.setText(airSlider.getValue()/1000.0f + "");
                horizontalDistanceField.setText("");
                maxHeightField.setText("");
                timeOfFlightField.setText("");

                //Clear graphs
                clearTrace();
            }else if("Close".equals(arg0.getActionCommand())){
                simulatorFrame.setVisible(false);
            }
        }

        public void stateChanged(ChangeEvent arg0) {
            if(velocitySlider.getValueIsAdjusting()){
                velocityField.setText(velocitySlider.getValue()/10.0f + "");
            }else if(angleSlider.getValueIsAdjusting()){
                angleField.setText(angleSlider.getValue() + "");
            }else if(gravitySlider.getValueIsAdjusting()){
                gravityField.setText(gravitySlider.getValue()/10.0f + "");
            }else if(airSlider.getValueIsAdjusting()){
                airField.setText(airSlider.getValue()/1000.0f + "");
            }
        }
}
```

Launch3DSimulator.java

```
package com.sim.launch;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
import javax.swing.JDesktopPane;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;

import com.sim.view.CircularMotionSimGUI;
import com.sim.view.EnergySimGUI;
import com.sim.view.ForceSimGUI;
import com.sim.view.PendulumMotionSimGUI;
import com.sim.view.ProjectileSimGUI;
import com.sim.view.common.SimGUI;
import com.sim.view.common.SimulatorGUIInterface;

public class Launch3DSimulator implements ActionListener
{

	private JDesktopPane	mainDesktop = null;
	private JFrame			mainFrame = null;

	/**
	 * Method will load Main application GUI
	 */
	public void loadMainGUI()
	{
		mainFrame = new JFrame("Mechanical Simulations");
		mainFrame.setBounds(10, 10, 1200, 740);
		mainFrame.addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent we)
			{
				System.exit(0);
			}
		});

		//Create the menu bar
		mainFrame.setJMenuBar(createMenuBar());

		mainDesktop = new JDesktopPane();
		mainFrame.getContentPane().add(mainDesktop,
BorderLayout.CENTER);
		mainFrame.setVisible(true);
	}

	/**
	 * Method will create menu bar and return to the calling function
	 * @return
	 */
	public JMenuBar createMenuBar()
	{
```

```
		JMenuBar simMenuBar = new JMenuBar();

		//Add simulation menu
		JMenu simulationsMenu = new JMenu("Simulations");
		simulationsMenu.add(SimGUI.cMenuItem("Energy",
KeyEvent.VK_E, this));
		simulationsMenu.add(SimGUI.cMenuItem("Force",  KeyEvent.VK_F,
this));
		simulationsMenu.add(SimGUI.cMenuItem("Pendulum Motion",
KeyEvent.VK_D, this));
		simulationsMenu.add(SimGUI.cMenuItem("Projectile Motion",
KeyEvent.VK_P, this));
		simulationsMenu.add(SimGUI.cMenuItem("Uniform Circular
Motion", KeyEvent.VK_C, this));
		simulationsMenu.add(SimGUI.cMenuItem("Exit",  KeyEvent.VK_X,
this));
		simMenuBar.add(simulationsMenu);

		return simMenuBar;
	}

	/**
	 * @param args
	 */
	public static void main(String[] args)
	{
		Launch3DSimulator sim3D = new Launch3DSimulator();
		sim3D.loadMainGUI();
	}

	/**
	 * Method will handle the action performed on the menu
	 */
	public void actionPerformed(ActionEvent arg0) {
		if("Energy".equals(arg0.getActionCommand())){
			SimulatorGUIInterface simInt = new EnergySimGUI();
			mainDesktop.add(simInt.loadMainGUI(mainFrame));
		}else if("Force".equals(arg0.getActionCommand())){
			SimulatorGUIInterface simInt = new ForceSimGUI();
			mainDesktop.add(simInt.loadMainGUI(mainFrame));
		}else if("Pendulum Motion".equals(arg0.getActionCommand())){
			SimulatorGUIInterface simInt = new
PendulumMotionSimGUI();
			mainDesktop.add(simInt.loadMainGUI(mainFrame));
		}else if("Projectile Motion".equals(arg0.getActionCommand())){
			SimulatorGUIInterface simInt = new ProjectileSimGUI();
			mainDesktop.add(simInt.loadMainGUI(mainFrame));
		}else if("Uniform Circular
Motion".equals(arg0.getActionCommand())){
```

```
                SimulatorGUIInterface simInt = new
CircularMotionSimGUI();
                mainDesktop.add(simInt.loadMainGUI(mainFrame));
            }else if("Exit".equals(arg0.getActionCommand())){
                System.exit(0);
            }
        }
}
```