DESINING AND IMPLEMENTING 3D, MULTIPLAYER AND REAL TIME COMPUTER GAME

(SUMMARY)

In this project, we have designed an infrastructure for 3D, multiplayer, real-time games and in the end of the studies a sample game is implemented by using the infrastructure. This document first gives brief information about 3D games programming, networking and multi-threading and then explains how to design and implement a computer game by combining these concepts.

Project is designed for Windows platform and all needs are provided by Microsoft's products and technologies. For 3D graphics operations DirectX (Microsoft DirectX SDK August 2006) and for network operations WinSock API are used. The C++ programming language is used on Microsoft Visual Studio 2005 IDE.

The project consists of four main libraries and two executable applications. The libraries have separated functions and can be used at any application without any change. These libraries handle mathematical (MathLib), graphics (GraphicsLib), object (ObjectLib), network (NetLib) operations. The executable applications are Server and Client applications which are based on the libraries. Here, server is a simple console application and client is a multi-threaded Win32 application. Besides these applications a helper application is written for designing game world easily. This helper application is written using C# 2.0.

At the end of this document, results of sample application are examined and some suggestions are stated according to these results.

ÜÇ BOYUTLU ve ÇOK OYUNCULU OYUN TASARIMI ve GERÇEKLEMESİ

(ÖZET)

Bu projede üç boyulu ve yerel ağ üzerinden çok oyunculu olarak oynanabilen herhangi bir bilgisayar oyununda kullanılabilecek bir alt yapı hazırlandı ve bu alt yapı kullanılarak örnek bir uygulama yapıldı. Bu dokümanda öncelikle 3 boyutlu oyun programlama, ağ iletişimi ve çok kanallı programlama hakkında öğrenilen kavramlar üzerinde duruldu ve daha sonra bu kavramların bir araya getirilerek 3 boyutlu ve çok oyunculu bir bilgisayar oyunun hazırlanışı hakkında bilgi verildi.

Yapılan çalışmanın oyun programlama üzerine seçilmesinin en önemli nedenleri, bilgisayar oyunlarının dünya üzerindeki yaygınlığının giderek artması ve nesneye dayalı programlamanın tam anlamıyla uygulanabileceği bir alan olmasıdır. Her geçen gün bilgisayar oyunlarının kalitesi ve bu oyunların yapımında kullanılan teknolojiler gelişmekte ve buna paralel olarak bilgisayar oyunlarının dünya pazarı üzerindeki yeri artmaktadır. Bu da bilgisayar oyunlarının yazılım alanında göz ardı edilemeyecek şekilde öne çıkmasını sağlamaktadır.

Uygulama tamamen nesneye dayalı programlama modeline göre modellenmiş ve gerçeklenmiştir. Oyun içerisindeki varlıklar ve işlemler çeşitli sınıflarla programatik olarak ifade edilmiştir. Programlama dili olarak nesneye dayalı özellikler barındıran C++ seçilmiştir. Bunun dışında oyun programlama da önemli bir nokta olan sistem kaynaklarına erişim ve dinamik bellek yönetimi konusunda güçlü bir dil olması da bu çalışmada C++'ın kullanılmasının ayrı bir sebebidir. Günümüzde ticari amaçlarla yazılan bir çok oyunda da gene bu dil kullanılmaktadır.

Geliştirilen proje, Windows platformuna yöneliktir ve hemen hemen bütün gereksinimler Microsoft ürünlerinden/teknolojilerinden karşılanmıştır. Projenin Windows platformuna yönelik seçilmesinin nedeni yine ticari amaçlarla yazılan bir çok bilgisayar oyununun bu platforma yönelik gerçekleştirilmesidir. Grafik işlemleri için Microsoft tarafından geliştirilen DirectX teknolojisi (Microsoft DirectX SDK August 2006), kullanılmıştır. Ağ haberleşmesi için WinSock API, çok kanallı programlama içinse "windows.h" kütüphanesi altında bulunan standart sistem fonksiyonları kullanılmıştır. Projedeki kodlar, Microsoft Visual Studio 2005 üzerinde derlenmiştir.

Proje esas olarak dört ana kütüphane ve iki çalıştırılabilir uygulamadan oluşmaktadır. Burada kütüphaneler birbirinden bağımsız olarak tasarlanmış olup her biri ayrı ayrı projelerde tek başlarına kullanabilecek şekilde tasarlanmıştır. Bu kütüphaneler MathLib, GraphicsLib, NetLib ve ObjectLib olarak isimlendirilmişlerdir. Çalıştırılabilir uygulamalar ise Server ve Client isimlendirilmiş olup, Server bir konsol uygulaması, Client ise Win32 uygulaması olarak yazılmıştır.

MathLib, oyun programlamada önemli bir yer tutan vektör ve matris işlemleri için hazır sınıf ve metotlar ihtiva eder. Bilindiği gibi oyun programlamada her nesnenin bulunduğu olduğu nokta, cisimler ve/veya oyun alanına bakan göz için ön, sağ ve üst yönlerini gösteren vektörler, dünya ve görüntü matrisleri şeklinde ifade edilirler. İşte bunların değişmesi gerektiğinde bazı karmaşık vektör/matris işlemleri yapılması gerekmektedir. Oyun içerisinde yapılacak bu işlerin MathLib kütüphanesi içerisindeki sınıflar tarafından yönetilmesi planlanmıştır. Ancak görülmüştür ki Microsoft DirectX SDK içerisinde yine aynı işleri sağlayacak yapı ve metotlar vardır ve bunlar MathLib kütüphanesinden daha performanslı çalışmaktadır. Bu nedenle eğer ihtiyaç duyulan bir işlem hem DirectX hem de MathLib tarafından karşılanabiliyorsa buralarda DirectX fonksiyonları tercih edilmiştir. Ayrıca Mathlib kütüphanesinde yazılan sınıflar DirectX içerisindeki yapılara özdeş olarak yazılmıştır. Bu da MathLib kütüphanesindeki sınıflara ait nesnelerin adresleri üzerinden DirectX yapılarına dönüştürülebilmesine imkan tanımaktadır. Bu kütüphane yazılan diğer kütüphanelerden bağımsız olarak tasarlanmıştır ve her hangi başka bir uygulamada hiç bir değişiklik yapılmadan kullanılabilir.

GraphicsLib'in esas amacı karmaşık Win32 ve grafik programlama içeriğini kapsüllemektir (encapsulation). Kütüphane içerisindeki Graphics sınıfında ağırlıklı olarak DirectX'in grafik işlemleri ile ilgili tarafı kullanılmıştır (Direct3D). Ekrana 3 boyutlu modeller çizdirilmesi, yazılar yazılması gibi işlerden sorumludur. Oyun içerisinde bu işlerden sorumlu sadece bir nesnenin bulunması yeterli ve gerek olduğundan bu sınıf singleton tasarım deseni kullanılarak gerçeklenmiştir. Kütüphane içerisindeki Form sınıfı karmaşık Win32 yapısını saklayan bir yapıdadır. Oyun içersindeki nesnelerin çizileceği pencereyi temsil eder. Yine oyun için tek bir pencerenin bulunması gerek ve yeter olduğundan bu sınıf da singleton tasarım desenine göre modellenmiştir. Bunların yanı sıra GraphicsLib içerisinde kullanıcıdan klavye ve fare aracılığı ile girdi almayı sağlayan sınıflar mevcuttur. Burada girdi alma işlemleri için hem standart Win32 fonksiyonları hem de DirectX içerisinde yer alan DirectInput API'si kullanılmıştır.

NetLib, oyun içerisindeki nesnelerin ağ üzerinden birbirleri ile haberleşmesini sağlayacak sınıfları ihtiva eder. Her varlık, bu kütüphane içerisindeki sınıfları kullanır. Böylece gerek sunucu gerekse oyuncular arasında ortak bir dil kurularak bunların birbirlerini anlayabilmesi sağlanmış olur. Oyunun istemci tarafında iki ana iş vardır: oyunu yönetip ekrana çizdirme işlemleri ve ağ haberleşmesi işlemleri. Oyun içerisindeki ağ haberleşmesinin ayrı bir iş parçacığı olarak tasarlanmıştır. Bu nedenle NetLib içerisinde çok kanallı programlama için gerekli sistem fonksiyonlarını kapsülleyen sınıflar mevcuttur. Bu sınıflar projenin istemci tarafında kullanılmıştır.

ObjectLib oyun içerisindeki varlıkların ve oyun alanın modellendiği kütüphanedir. Bunların yanı sıra üç boyutlu modelleri (mesh), yüzey kaplamalarını (texture), cisimlerin durumlarını ve oyuna bakışı (camera) modelleyen başka sınıflar da mevcuttur. Bu kütüphane içerisinde nesneler ortak bir sınıftan türetilmiştir. Daha sonra türetme ile sınıflar giderek özelleşmiş ve oyun içerisindeki varlıklara veya oyun alanına dönüşmüştür. Oyun alanı hücrelere bölünerek yönetilmiştir. Böylelikle uygulamanın performansında artış sağlanmaya çalışılmıştır. Örneğin, oyun içerisinde çarpışmalar kontrol edilirken sadece aynı hücre içerisinde cisimler arasındaki çarpışmalar kontrol edilmiştir. Nesneler arasındaki iletişim ve etkileşim bu kütüphane içerisinde bulunan yönetici sınıf tarafından kontrol edilmektedir. Bu kütüphane diğer kütüphanelerden bağımsız olarak tasarlanmıştır ve DirectX'in kullanıldığı her hangi bir projede değiştirilmeden kullanılabilir.

Kütüphanelerin yanı sıra proje içerisinde iki adet çalıştırılabilir uygulama mevcuttur, bunlardan birisi sunucu (Server) diğeri ise istemci (Client) uygulamadır. Oyun içerisinde istemciler sunucu üzerinden birbirleriyle haberleşmektedir. Hazırlanan alt yapı gerçek zamanlı bir uygulama için tasarlandığından iletilen paketlerin mümkün olduğunca çabuk iletilmesi gerekmektedir. Ayrıca yine gerçek zamanlı uygulamalarda bir paketin geç iletilmesi yerine hiç iletilmemesi daha verimli olmaktadır. Bu nedenle uygulamada haberleşme standardı olarak UDP/IP protokolü tercih edilmiştir. Sunucu varsayılan olarak 20000, istemci ise 20001 numaralı portu dinlemektedir. Bu nedenle ilgili uygulamanın çalıştırılacağı bilgisayarda (varsa) güvenlik duvarının bu portlardan gelen paketlere açık olması gerekmektedir.

Sunucu uygulama basit bir konsol uygulaması şeklindedir. Görevi ise kendisine gelen mesajlara yanıt vermek veya bu mesajları diğer istemcilere göndermektir. Bu noktada sunucunun hangi paket karşısında nasıl davranacağını bilmesi için NetLib kütüphanesi içerisinde tanımlı mesaj formatını tanıması gereklidir.

İstemci uygulama ise Win32 uygulaması olarak tasarlanmıştır. İstemcinin gerçekleştirdiği üç ana işlem vardır. Bunlardan birincisi oyunu yönetmek ve oyun içerisindeki etkileşimlere göre ekranın çizdirilmesini sağlamaktır. Oyun isteğe göre, bir pencere içerisinde ve ya tam ekran olarak oynanabilir. Diğer iki işlem ise ağ haberleşmesi ile ilgilidir. Birinci işlem yerel bilgisayardaki oyucunun durumunda bir değişiklik olduğunda, yani hızı, konumu, yönü, görünüşü vs. değiştiği zaman bu bilgiyi ağdan sunucuya iletmektedir. İkinci işlem ise sunucuyu dinleyerek oyun içerisindeki diğer oyuncuların durumlarını güncellemektedir. İstemci uygulamada performansı arttırmak için bu üç işlem ayrı iş parçacıkları olarak tasarlanmıştır. Yani istemci tarafta çok kanallı programlamaya gidilmiştir.

Bunların dışında uygulama içerisindeki istisnai durumları yakalamak ve dosya işlemlerini kontrol etmek için bazı yardımcı sınıflarda mevcutlarda yazılmıştır. Bunlar kütüphane haline getirilmeden ilgili yerlerde çağırılarak kullanılmıştır.

Buraya kadar hazırlanan alt yapının üzerine örnek bir oyun uygulaması geliştirilerek hazırlanan alt yapı test edildi. Hazırlanan örnek uygulamanın çözünürlüğü 640x480 piksel olarak seçilmiştir. Oyun içinde ana bir döngü vardır. Bu ana döngü:

1. Kullanıcıdan klavye/fare girdisini alır.
2. Nesneleri hareket ettirir ve çarpışmalardan doğan sonuçları değerlendirir.
3. Bütün hareketler tamamlandıktan sonra kamera noktası ve bakış açısı oyuncuya göre ayarlanır ve ekran çizdirilir.

Oyunda ekranın çizdirilme hızı 66 (kare/sn) olarak sabitlenmeye çalışılmıştır. Bunun için ana döngü 15 ms.'den kısa sürüyorsa belli bir süre bekleme konulmuştur.

Geliştirilen oyun basit olarak bir yakalamaca oyunu şeklindedir. Oyun içerisinde bir adet ebe vardır bu oyuncunun rengi (kırmızı) diğer oyuncularınkinden (yeşil) farklıdır. Ebe diğer oyunculardan birine dokunduğunda artık o oyuncu ebe olur ve renkler değişir. Oyun kapalı bir labirent içerisinde oynanmaktadır. Oyun aynı anda en çok on adet oyuncu tarafından oynanabilir.

Bunlarla beraber proje sırasında teorik olarak çalışılan ancak uygulama içerisine konulmayan iki önemli konu vardır. Bunlardan ilki 'ikili uzay bölümleme' (BSP – Binary Space Partitioning) algoritmasıdır. Bu algoritma sayesinde oyun alanındaki hangi

hücrelerin görüleceğini önceden kestirilerek çizdirme işlemi hızlandırılabilir. İkinci konu ise ‘gölge çizimidir’. Gölge çizdirme işlemi ise oyun içindeki nesnelerin ışığa göre konumlarına bağlı olarak gerçekçi bir gölge çizdirme işleminin yapılmasıdır. Burada ekran bir kere göz ışık kaynağının olduğu yerdeymiş gibi bir kere çizdirilir ve nerelerin gölge olacağı ortaya çıkar. Daha sonra ekran bir kere de normal çizdirilir. Ekran iki kere çizdirildiği için bu işlem uygulamanın performansını düşürmektedir.

Ayrıca oyun yapımından ayrı olarak oyun içerisindeki haritanın oluşturulmasında kolaylık sağlaması açısından yardımcı bir uygulama hazırlandı. Bu uygulamada programlama dili olarak C# 2.0 kullanıldı.