T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

SOSYAL AĞ OYUNU TASARIMI VE GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Hazırlayan
Alper YILDIRIM

Danışman
Yrd. Doç. Dr. Oğuz KARAN

İstanbul – 2012

ÖNSÖZ

Yüksek lisans eğitimim ve tez çalışmamın tamamlanması süresince büyük bir gayret ve özveriyle çalışmamı takip eden, gösterdiği sabır ve hoşgörüyle bana destek olan tez danışmanım Sayın Yrd. Doç. Dr. Oğuz KARAN'a çok teşekkür ederim.

Proje boyunca bana destek olan eşime ve bu günlere gelmemde büyük pay sahibi olan aileme teşekkürlerimi sunarım.

HAZİRAN 2012

Alper YILDIRIM

İÇİNDEKİLER

SayfaNo

KISALTMALAR

GPU : Graphics processing unit

FPS : Frame per second

ŞEKİL LİSTESİ

SayfaNo

TABLO LİSTESİ

SayfaNo

ÖZET

SOSYAL AĞ OYUNU TASARIMI VE GELİŞTİRİLMESİ

Bu projede facebook üzerinde çalışan ve en son teknolojileri kullanan sosyal bir oyun uygulaması geliştirilmiştir. Oyun içinde alışılageldik 2 boyut yerine 3 boyut kullanılmıştır.

Oyuncu oyun içerisindeki marketten çeşitli evler ya da tarlalar satın alarak kendi şehrini kurmaya çalışacaktır. Boş tarlalara ürün ekerek ya da evlerini kiraya vererek para ve deneyim kazanabilir. Ancak oyuncu, tarlalarıyla düzenli olarak ilgilenmezse ektiği ürünler çürüyebilir ve para ya da deneyim kazanamaz. Bu durumda sadece yeni ürünler ekmek üzere tarlasını temizleyebilir. Oyuncunun amacı, daha fazla para kazanarak şehrini güzelleştirmek ve büyütmektir. Kazandığı paralarla daha lüks evler almaya çalışır. Bu durumda daha fazla kira toplayabilir. Aynı şekilde tarlaları içinde daha fazla para kazandırabilecek ürünler ekebilir. Daha yeni evler dikebilmek için eski evlerinin ya da tarlalarının yerini değiştirebilir veya tamamen haritadan silebilir. Bir objenin yerini değiştirdiğinde kazanacağı para ya da deneyimde bir değişiklik olmaz. Bu para ve deneyimler zamana bağlı olarak kazanılır. Oyuncu bir eve kira alabilmek için etkileşimde bulunduğunda süre dolmamışsa kirasını alamaz. Benzer şekilde tarlasını hasat edemez.

Oyuncu, tarlasına ürün ekmek, tarlasını hasat etmek, çürümüş tarlasını temizlemek, bir evi kiraya vermek, marketten bina ya da tarla satın almak istediğinde birer aksiyon oluşur. Bu aksiyonlar sonrasında yapılmak istenen eylemin mantıklı olup olmadığı kontrol edilmektedir. Eğer mantıklı ise client serverı protokoller yardımıyla bilgilendirir. Bu protokolleri server açarak gelen bilgilerin doğruluğunu tekrar kontrol eder. Eğer bilgiler doğru ise veri tabanında gerekli değişiklikleri yapar ve Flash'a geri bildirim gönderir.

Bilgileri hem server(.net) hem de client(Flash) tarafının kontrol etmesinin amacı kötü niyetli kişilerin oyuna müdahale etmesini engellemektir. Eğer server gelen bilgileri kontrol etmeden veri tabanında gerekli değişiklikleri yaparsa kullanıcı client ile server arasında gidip gelen protokolleri dinleyebilir ve bu protokolleri simule edebilir. Bunun sonucunda paranın güncellenmesini serverdan talep ederse server bu değişikliği mantıklı olmasa bile veri tabanında yapabilir. Bu gibi haksızlıkları önlemek için server'ın bilgileri kontrol etmesi gerekir.

Flash tarafının bilgileri kontrol ederek yollamasının amacı ise server tarafından olumsuz dönebilecek cevapları server'a göndermeden kendi içinde tespit ederek server yükünü azaltmaya çalışmaktır. Milyonlarca insanın oynadığı bir oyunda her oyuncunun her hareketini server'ın kontrol etmeye çalışması server için hamallıktır. Bu yükü hafifletmek amacıyla client tarafında da gerekli kontrollerin yapılması gerekir.

Anahtar Kelimeler: Facebook, sosyal oyun, sosyal ağ, flash

SUMMARY

SOCIAL NETWORK GAME DESIGN AND DEVELOPMENT

In this project, a social game application, which is using recent technologies, runs on the facebook, is developed. In the game, 3D is used instead of standard 2D.

Player tries to establish his/her own city by buying varied houses or fields from the market in the game. Player can earn many and bonus by planting crop or renting his/her houses. But, if player doesn't take care his/her cultivated lands regularly, the crops can molder and player cannot earn money or bonus. In this situation player can only purified lands for planting new crops. The aim of the player is to beautify and to grow his/her city by earning more money. With money player tries to by luxury houses so that player can collect more many from renting. Same as the houses, player can plant a crop which makes his/her earn more money. To built new houses player can change the places of the houses fields or can totally delete from the map. To change the places of the object don't create any changes in the level of many or bonus. This money and bonus are earned by depending on time. If the time of the rented houses has not filled yet, player cannot earn money when he/she tries to interact with the rented houses. Same as the houses crops cannot be cultivated before the time is up.

When the player wants to plant a crop, cultivate the crops, purify the fields, rent the houses by building or fields from market there a merge an action. After these actions the rationality of the action is controlled if the action is rational server can be informed by the help of protocols. Server checks the truth of the knowledge again by opening these protocols. If the knowledge is correct the sufficient changes are madden in the database and are sent feedback to the flash.

The aim of the control of the knowledge by both server (aspx) and client (flash) is to prevent intervention to the game by people in bad faith. If the server makes the sufficient changes in the database without checking the knowledge, user can listen the protocols. Which are in between server and client and user demands to update the money, server make this changes in the database even the knowledge is not rational. To prevent this cheating, server has to check to knowledge.

Flash sends the knowledge by checking them. The aim of the flash for doing this checking is to not fire out server's effort needlessly. It is meaningless for server to check every single action of the every player in any games which are played by millions of people. To mitigate the burden of the server client has to make sufficient controls.

Keywords: Facebook, social game, social network, flash

1. GİRİŞ ve AMAÇ

Web tabanlı oyunlar, oyuncuların herhangi bir yükleme yapmadan doğrudan oynayabilecekleri oyunlardır. İnternet üzerinden oynanırlar. Dolayısıyla bu tarz oyunlar internet erişimi olan heryerden oynanabilmektedir. Bu tarz oyunlar tek oyunculu ya da çok oyunculu olarak oynanabilir. Tek oyunculu oyunlar oyuncunun bilgisayarla oynadığı oyunlardır. Bilgisayara oyunla ilgili yapay zeka programlanır. Bilgisayar oyuncunun hamlelerine karşılık vererek oyunu kazanır yada kaybeder. Bilgisayar bu hamlelere o oyun için çıkartılmış oyun ağacına bakarak karar verir. Çok oyunculu oyunlar ise oyuncuların birbirleriyle tam etkileşimde olduğu oyunlardır. Bu tarz oyunları oyuncular birbirlerine karşı oynarlar. Facebook'un sosyal ortamı, kullanıcı sayısının çokluğu ve üçüncü şahıslara sunduğu uygulama geliştirme altyapısı oyun firmalarının dikkatini çekmiştir. Sosyal ağlar için oyun üreten bu firmalardan bazıları milyonlarca kullanıcıya ulaşmayı başarabilmişlerdir. (Ben Kirman, 2009)

Bu çalışmada baştan sonra bir Facebook üzerinde çalışan sosyal oyun uygulamasının nasıl yapılacağı anlatılmıştır. Alışılageldik *facebook* oyunlarından farklı bir çalışma olmasını sağlamak amacıyla 2012 yılı ilk çeyreğinde beta statüsünden çıkan *flash player 11* ile beraber gelen stage3D teknolojisi oyunda kullanılmıştır. Bu teknoloji flash'ın gpu kullanabilmesine ve oyunun 3 boyutlu olmasına olanak sağlamıştır.

1.1. Benzer Çalışmalar

Sosyal ağların üye sayılarındaki hızlı yükselişi Tablo1.1 ve Tablo 1.2 de görülmektedir. Sosyal ağların 3. şahıslara uygulama geliştirme olanağı sağlaması yepyeni bir oyun türü olan sosyal ağ oyunlarının doğmasına neden olmuştur. Sosyal ağlar gibi bu ağlar için geliştirilen oyunlarda kullanıcı sayısı artışı bakımından başarılı bir yükseliş göstermişlerdir.

Tablo 1.1: Sosyal Ağların Kullanım Artış İstatistikleri

Total Worldwide Home/Work Locations Among 15 Yaş Üstü İnternet Kullanıcıları (2007 vs 2006) Kaynak: comScore World Metrix			
Sosyal Ağ Sitesi	**Toplam Tekil Ziyaretçi (000)**		
	Haz-06	**Haz-07**	**% Değişim**
MySpace	66,401	114,147	72
Hi5	18,098	28,174	56
Friendster	14,917	24,675	65

Tablo 1.2: Sosyal Ağların Günlük Kullanım İstatistikleri

Total Worldwide Home/Work Locations Among 15 Yaş Üstü İnternet Kullanıcıları (2007 vs 2006) Kaynak: comScore World Metrix			
Sosyal Ağ Sitesi	**Ortalama Günlük Ziyaretçi (000)**		
	Haz-06	**Haz-07**	**% Değişim**
MySpace	16,764	28,786	72
Hi5	3,742	14,917	299
Friendster	3,037	5,966	96

Facebook'un sosyal ortamı sayesinde daha hızlı yayılma imkanı bulan bazı oyun firmaları çok kısa sürede milyonlarca oyuncuya ulaşabilme başarısını göstermişlerdir. Bu firmalar arasında en ünlüsü olan *Zynga* 60 milyon günlük aktif kullanıcı sayısına (DAU) ve 232 milyon aylık aktif kullanıcı sayısına (MAU) ulaşmış bulunmaktadır. 60 milyon kullanıcı her gün 416 milyon kez birbiri ile etkileşime geçmektedir. (Independent, Accurate Application Metrics and Trends from Inside Network, 2012)

Facebook'un avantajlarını iyi kullanan Zynga 2010 yılında sanal ürün satışından ve reklamdan 839 milyon dolar gelir elde ettiği bilinmektedir.

2009 yılında bu değer 328 milyon dolar, 2008′de ise 36 milyon dolar iken sadece 4 yılda 1,5 milyar doları aşan gelir elde eden *Zynga* konunun önemi hakkında önemli ipuçları sunmaktadır.

2014 yılında sosyal ağlardaki kullanıcı sayısının 1,6 milyara ulaşacağı öngörülmektedir. Bunun yanında iş modelleri olan ücretsiz oyna ve oyun içi sanal ürün satışı modelinin de doğru olduğunu ve büyüyeceğini belirtilmektedir. Bir araştırmaya göre 2010 yılında tüm oyun endüstrisinde oyun içi sanal ürün satışı 7,3 milyar dolar olarak gerçekleşirken bu değerin 2014′te ikiye katlanması tahmin edilmektedir. (Reddy, 2009)

Online oyunların tarihine bakacak olursak sosyal ağlara entegre olunmadan önce bile bu oyunların çok geniş kitlelere ulaşabildiği görülmektedir. 2000 yılında Oslo'da Fifth Season AS Gurubunun Programladığı İngilizce oyun Planetarion modern Web tabanlı oyunların atası sayılabilir. İlk sene içerisinde 175.000 kayıtlı oyuncuyla zamanında en fazla oyuncu sayısına ulaşmıştır. Ücretli hesapların başlamasıyla çoğu oyuncular başka bir oyun ararken Galaxywars oyunu ortaya çıkmış ve 100000 oyuncu sayısına kadar ulaşmıştır. Profesyonel yönetimi yetersiz olduğundan eski oyuncular aralarında toplanarak kendi oyunlarını geliştirmişler ve bu nedenle Alman web tabanlı oyun sektöründe çok sayıda birbirine benzeyen oyunlar bulunmaktadır, bu oyunlardan bazıları günümüze kadar başarıyla ayakta kalmayı başarabilmişlerdir.

3 Ekim 2002 tarihinde *OGame* oyunu ilk Evreni açılmış ve şuan Avrupa da en büyük Web tabanlı olduğu düşünülmektedir. Çok başarılı olan bu oyunun Programcıları *Klaas Kersting* ve *Alexander Rösner* Aralık 2003 (bir sene sonra) *Gameforge GmbH* şirketini *Karlsruhe'*de kurdular. Web tabanlı oyunlardan hariç 3 boyutlu *client* oyunları yapan *Gameforg*e AG şuan toplam 20 oyun ve 54 dilde online oyun sektöründe ilk on sıralarda yer almaktadır. *Bigpoint* 50 çevrimiçi oyunu ve 20 dilde 99 mil kayıtlı oyuncusuyla dünya çapında oyun portal sayfalarında ilk 3 arasında bulunmaktadır. *Innogames* ve *Travian* web tabanlı çevrimiçi sektöründe tanınmış şirketler arasında yer almaktadırlar.

Şuanda yerli firmalardan Consala Games Sonkorsan, Novagun ve Minefight oyunlarını facebook'a entegre ederek bir milyon kayıtlı kullanıcı sayısını aşmış durumda.

2. Halicville Uygulamasında Kullanılan Teknolojiler

Çalışma boyunca 3 farklı programlama dili ve birçok yardımcı yazılım kullanılmıştır.

2.1. Flash

Flash başlangıçta *Future Splash* Animatör olarak işe başlangıç yapmıştır. Bu programın amacı vektörel görüntüler yaratmak ve bu görüntüler üzerinde hareketlendirmeler yapmaktır. Macromedia firması *Future Splash'ı* satın almış ve ismini de Flash olarak değiştirmiştir. Bu programı da *World Wide Web* için grafiksel içerik oluşturan bir araç olarak tanıtmıştır. Flash programı vektörlere daha doğrudan ulaşmamıza imkân sağlar. Yeni başlayanlar için basit hareketli görüntüler yaratmasına imkân sağlar. Flash animasyon yaratmak için sıradan animasyon metotlarını kullanır. Her biri bir sonrakinden çok az farklı görüntülerin birleştirilmesi mantığı Flash içinde geçerlidir. Aradaki hassasiyeti ayarlama işi de tamamen kullanıcıya bırakılmıştır. Sadece görüntü değil seslerinde animasyonlarınızı yaratma imkânı sağlamaktadır. Flash, yarattığınız animasyonların bir bilgisayar ya da Web tarayıcısında izlenilebilir kılınmasını sağlayan bir oynatım sistemidir. Yarattığınız animasyonların uzantısı ".fla" olmaktadır. İzlenebilen filmler yaratmak için Flash Player formatına dönüştürmek gerekir. Flash Player formatına çevrilen dosyalarda ".swf" uzantısına sahip olmaktadır.

Yazılımın Genel Özellikleri:

Çok güçlü bir programlama diline (Action Script 3) sahiptir. Bu dil ile web servisi, socket vb bağlantılar kurmaktan görüntü işlemeye hatta 3d yazılımlar yapabilmeye olanak sağlamaktadır.

Flash player 11 ile beraber gpu desteği kazanmıştır.

Serbest çizim aracı ve gelişmiş bir flash düzenleme takımı ile her türden çizim yapılabilir.

Geliştirilen oyunlar için hızlandırma ayarlarına sahiptir.

Web siteleri için etkileşimli menüler ve butonlar hazırlanabilir.

Kendiliğinden gölgelendirme, doku ve eğim ayarlama, üç boyutlu efektler verme gibi serbest seçim araçları vardır. Flash animasyon ve oyun hazırlamak için her türden araca sahiptir.

2.1.1. Neden Flash?

Web ortamında herhangi bir video dosyası tarayıcıya yükleninceye kadar hayli bir zaman harcanır, çoğu zamanda bilgisayarın veya tarayıcının kilitlenmesine sebep olur. Aynı tür sorunlar yüksek çözünürlükte resim dosyalarının açılmasında da yaşanmaktadır.

Flash animasyonlarının tercih sebeplerinden en önemlisi web ortamında uygun, küçük dosya boyutlarını işgal etmesidir. Flash'ın bu denli küçük dosya boyutları ile büyük işler başarmasının sırrı, programın vektörel bir taban üzerinde çalışmasından kaynaklanmaktadır.

Bilgisayar verileri bellekte veya disk alanında saklarken veriye ait değerleri dosya içerisinde tutar. Örneğin; bir bitmap resmi disk alanında saklandığında bu resme ait her nokta (pixel), noktaların yerleri, noktalara ait renkler, dosyanın boyutunu belirmektedir. Resmin çözünürlüğü arttıkça diskteki alanda o denli artacaktır. Oysaki, vektörel animasyonda verilere ait her noktanın saklanmasına gerek yoktur. Sadece koordinat düzleminde konum ve büyüklük değerlerinin tutulması yeterli olmaktadır. Bu da, Flash animasyonlarının web ortamında hızlı çalışmasını, bulunduğu ortamda az bir alan kaplamasını sağlamıştır.

2.2. C#

C#.NET, Microsoft'un güçlü, bileşen yönelimli dilidir. C#, Microsoft .Net Framework mimarisinde önemli bir rol oynamakta ve kimileri bu rolü C' nin UNIX' in gelişiminde oynadığı rolle karşılaştırmaktadırlar. C#, kuşak imleri kullanır; bu yüzden zaten C, C++ ya da Java gibi bir kuşak imi dili bilenler için öğrenmesi daha kolaydır. (John Sharp, 2005)

Web uygulamaları oluşturan ve çalıştıran bütün yapıların karşılaması gereken gereklilikler vardır.

- HTTP protokolünü desteklemelidir.
- İstemci durumunu verimli biçimde yönetebilmektedir.
- Web uygulamalarının kolay geliştirilebilmelerini sağlayan araçlar sunmalıdır.
- HTML' i destekleyen bütün tarayıcıların erişebilecekleri uygulamalar geliştirmelidir.
- Yanıt veren ve ölçeklenebilir olmalıdır.

Bu gereksinimlerin birçoğunu karşılamak için Microsoft, Active Server Pages modelini geliştirmiştir. Active Server Pages, program geliştiricilerin uygulama kodunu HTML sayfalarına eklenmesi sağlamıştır. Microsoft Internet Information Services (IIS) gibi bir Web sunucusu uygulama kodunu çalıştırabilir ve onu bir HTML yanıtı oluşturmak için kullanabilir.

.NET platformu geliştirilirken Microsoft, Active Server Pages yapısını güncelleştirdi ve ASP.NET' i yaratmıştır. ASP.NET'in ana özellikleri şunlardır:

- İş mantığını ayrıştıran sunum mantığı ve kod dosyaları içeren Web formlarını kullanan gerçekçi bir program modelidir. Kodu, C# da dahil olmak üzere desteklenen herhangi bir .NET dili ile yazılabilir. ASP.NET Web formları derlenir ve başarı arttırmak için Web sunucu üzerinde önbelleğe kaydedilir.
- Sunucu olaylarını destekleyen ama bütün HTML uyumlu tarayıcılarda doğru biçimde işlenmelerini sağlamak için HTML haline getirilen sunucu denetimleri. Microsoft ayrıca, kodunuzun içinde düzenlemenizi sağlayacak biçimde, birçok standart HTML denetimi de geliştirilmiştir.
- Bir ADO.NET veri kaynağından alınan verileri görüntülemek, düzenleme ve bakımını yapmak için güçlü veri denetimleri.
- Tanımlama bilgileri kullanarak istemci durumunu istemci bilgisayarda, özel bir serviste (ASP.NET State servisi) Web sunucusu üstünde ya da bir Microsoft SQL Server veri tabanında önbelleğe almak için seçenekler. Önbelleğe alınan bu durum bilgileri, kod kullanarak kolayca programlanabilir.

2.2.1. Neden C#?

ASP.Net bize programlama dillerinde esneklik sağlamıştır. Bunun içindir ki Microsoft Visual Studio bünyesinde bulunan dillerden herhangi birini bilmemiz projemize başlayabilmemiz için yeterli olacaktır.

Microsoft Visual Studio'nun bize sağlamış olduğu zengin araç içeriği ile hem göze hitap eden hem de hızlı projeler hazırlayabiliriz ki bu sadece bizim hayal gücümüzle sınırlıdır.

Asp.Net gerçek programlama dillerini kullandığı için kodlarımızı istediğimiz gibi derleyebilmemize olanak sağlamaktadır. Özellikle gerçek verilerin kullanılması (*integers* ve *strings*) ve durum işleme sayesinde mantıksal ilişkiler kurabilmemizi ve bunu kullanabilmemizi sağlıyor.

C# Programlama Dili, Microsoft'un geliştirmiş olduğu yeni nesil dilidir. Yine Microsoft tarafından geliştirilmiş ".NET" Teknolojisi için geliştirilmiş dillerden biridir.

Microsoft tarafından geliştirilmiş olsa da ECMA ve ISO standartları altına alınmıştır.

C programlama dilinde bir tamsayı değişkeni 1 attırmak için ++ soneki kullanılır. C++ dili adını, C diliyle Nesneye Yönelimli Programlama yapabilmek için eklentiler (C With Classes) almıştır. Benzer şekilde C++ diline yeni eklentiler yapılarak ((C++)++) bir adım daha ileriye götürülmüş ve tamamen nesneye yönelik tasarlanmış C# dilinin isimlendirilmesinde, + karakterlerinin birbirlerine yakınlaşmış hali ve bir melodi anahtarı olan C# Major kullanılmıştır.

Bu dilin tasarlanmasına Pascal, Delphi derleyicileri ve J++ programlama dilinin tasarımlarıyla bilinen Anders Hejlsberg liderlik etmiştir.

Birçok alanda Java'yı kendisine örnek alır ve C# da java gibi C ve C++ kod sözdizimine benzer bir kod yapısındadır. .NET kütüphanelerini kullanmak amacıyla yazılan programların çalıştığı bilgisayarlarda uyumlu bir kütüphanenin ve yorumlayıcının bulunması gereklidir. Bu, Microsoft'un .Net Framewok'u olabileceği gibi ECMA standartlarına uygun herhangi bir kütüphane ve yorumlayıcı de olabilir. Yaygın diğer kütüphanelere örnek olarak Portable.Net ve Mono verilebilir.

Özellikle nesne yönelimli programlama kavramının gelişmesine katkıda bulunan en aktif programlama dillerinden biridir. .NET platformunun anadili olduğu bazı kesimler tarafından kabul görse de bazıları bunun doğru olmadığını savunur.

C#, .NET orta seviyeli programlama dillerindendir. Yani hem makine diline hem de insan algısına eşit seviyededir. Buradaki orta ifadesi dilin gücünü değil makine dili ile günlük konuşma diline olan mesafesini göstermektedir. Örneğin; Visual Basic .NET (VB.NET) yüksek seviyeli bir dildir. Dersek bu dilin insanların günlük yaşantılarında konuşma biçimine yakın şekilde yazıldığını ifade etmektedir. Dolayısı ile buradan yola çıkarak VB.NET, C#.NET'ten daha güçlü bir dildir diyemeyiz. Programın çalışması gereken bilgisayarlarda framework kurulu olması gerekmektedir.

2.3. PHP

Php, facebook ile halicville uygulaması arasında bağlantı kurmak için kullanılacaktır. Arkadaş daveti hediye gönderme uygulama paylaşımları facebook kullanıcısından gerekli izinlerin alınması gibi etkileşimler php tarafından sağlanacaktır.

PHP (Hypertext Preprocessor) geniş bir kitle tarafından kullanılan, özellikle sanal yöreler üzerinde geliştirme için tasarlanmış HTML içine gömülebilen bir betik dilidir.

Bir PHP betiğinin Perl ya da C gibi dillerden ne kadar farklı bir yapıda olduğuna dikkat etmek gerekir. PHP kodu <?php ve ?> özel başlangıç ve bitiş etiketleri arasına yazılır. Bu etiketler "PHP kipine" rahatlıkla girip çıkabilmenizi sağlarlar.

PHP'yi Javascript gibi kullanıcı tarafında çalışan dillerden ayıran, sunucu tarafında çalıştırılıyor olmasıdır. PHP kullanmanın en güzel yanlarından biri, yeni kullanıcılar için öğreniminin oldukça kolay olması ve aynı zamanda profesyonel kullanıcılar için ileri seviyede özellikler içermesidir.

2.4. Microsoft SQL Server

SQL Server 2008, Microsoft'un Veri Platformu vizyonunu çerçevesinde, SQL Server 2008 R2, her an ve her yerde her türlü verinin yönetilmesine olanak verir. Doğrudan veri tabanı içinde yapılanmış ve yarı yapılanmış belgelerin yanında resim ve zengin medya gibi yapılanmamış belgelerden gelen verileri depolar. SQL Server 2008 veri-

lerin sorgu, arama, senkronizasyon, raporlama ve analiz gibi daha fazla işlem gerçekleştirilmesini sağlayan zengin bir entegre hizmetler seti sunmaktadır. (GÖZÜDELİ, 2011)

SQL Server 2008, kritik uygulamalar için en yüksek güvenlik, güvenilirlik ve ölçeklenebilirlik düzeylerini sağlamaktadır. Bugünün hızlı gelişen iş dünyasında yeni fırsatlardan faydalanmak için şirketlerin veri odaklı çözümleri hızlı bir şekilde oluşturabilmesi ve uygulayabilmesi gerekir. SQL Server 2008, uygulamaların yönetim ve geliştirme süresini ve maliyetini azaltmaktadır.

2.4.1. SQL Server Integration Services

SQL Server 7.0'dan bu yana kullanılagelen Data Transformation Services SQL Server 2005'den itibaren yeniden şekillendirilerek SSIS adı verilmiştir. SSIS ile veri bütünleştirme çözümleri geliştirmede kullanılmaktadır. SSIS veri birleştirme ve dönüşüm paketleri oluşturmak için grafik ara yüzler, sihirbazlar ve araçlar içermektedir. Veri akışı paketleri programlarken, FTP ile veri çekme ve ver gönderme, e-mail mesajları gönderme gibi ek destekler sağlamaktadır. Ayrıca SSIS bileşenlerine, SMO gibi programsal olarak erişmek mümkündür. (Henderson, 2003)

2.4.2. Saklı Yordam

Bir transact-SQL depolanmış yordamı, SQL Server veri tabanında depolanan ve kullanım sırasında derlenen bir T-SQL kod kümesidir. Bu kod kümesi CREATE PROCEDURE komutuyla oluşturulur. Transact-SQL komutlarının çoğu depolanmış yordamlarda kullanılabilir. Ancak bazı komutlar (CREATE PROCEDURE, CREATE VİEW, SET SHOWPLAN_TEXT, SHOWPLAN_ALL gibi) bir komut toplu işindeki ilk (veya tek) deyim olmak zorundadırlar ve bu yüzden depolanmış yordamlarda kullanılamazlar. Transact-SQL komutlarının çoğu depolanmış yordamlarda, komut toplu işlerinde (command batch) davrandıkları gibi davranırlar. Ancak bazıları depolanmış yordamlarda kullanıldıklarında özel yetenekler kazanırlar veya farklı davranışlar gösterirler. (Henderson, 2003)

2.5. 3D Studio Max ve PhotoShop Yazılımı

3D Studio Max, Autodesk tarafından geliştirilen bir 3D modelleme, Görselleştirme ve Animasyon programıdır. MSDOS ortamında çalışan 3D Studio yazılımının devamı olan 3D Studio Max'in son sürümü, 2011 yılında çıkan 3DS Max 2012'dir.

Gelişmiş eklenti desteği ve kolay kullanımı ile 3DS Max, 3D modelleme yazılımları arasında en yaygın kullanıma sahip uygulamalardan biridir. Gelişmiş karakter modelleme özellikleri ile oyun geliştiricilerinin gözdesi haline gelmiştir. Film özel efektleri, mimari sunumlar ve endüstriyel tasarım sunumları gibi alanlarda da yaygın olarak kullanılmaktadır.

3DS Max, parçacık sistemleri, karakter modelleme araçları, hareket yakalama araçları ve gelişmiş denetçiler gibi özellikleriyle tek bir pakette çok sayıda özelliği sunmaktadır. Ayrıca MAXScript adında tümleşik bir programlama dili vardır.

3DS Max çok sayıda temel nesneyi hazır olarak sunar. Mimari tasarımlar için de duvar, kapı, pencere ve merdiven gibi bileşenleri ölçülerini kolayca değiştirerek projeye eklemek mümkündür. 3DS Max ayrıca poligonal modelleme, NURBS modelleme, yüzey modelleme gibi teknikleri destekler.

3. Halicville Uygulamasına Genel Bakış

Oyuncu, internet üzerinden oyuna bağlandığında şekil 3.1'deki gibi bir ekranla karşılaşır. Şekil3.2'de görüldüğü üzere üst menüden para ve deneyim bilgisinin yanı sıra oyunun kaç fps ile çalıştığını da görebilir. Sağ alt köşedeki düğmeler yardımıyla bir objeyi taşıyabilir, silebilir ya da market sayfasına ulaşabilir.

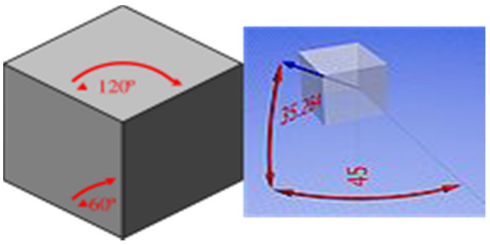
Şekil 3.1: Oyundan Bir Kare

Ekranın merkezinde ise satın aldığı objeleri görebilir. Eğer evin kira zamanı ya da tarlanın hasat zamanı dolmuşsa para simgesi belirir. Oyuncu bu objelerle etkileşimde bulunarak evin kirasını toplayabilir ya da tarlasını hasat ederek boş tarla görünümüne dönüştürebilir.

Şekil 3.2: Market Sayfası

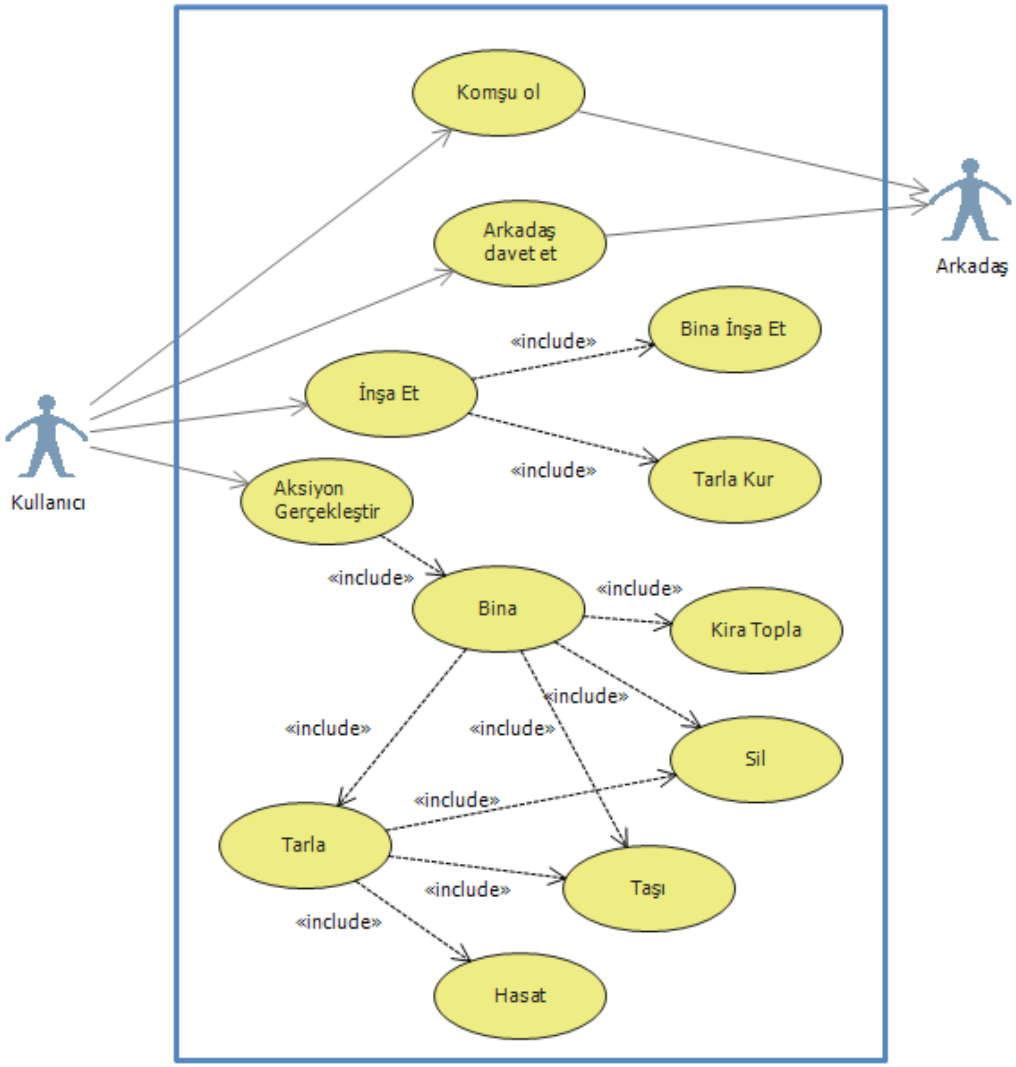
Oyuncu, market ekranında ilgili sekmeleri bu sekmeler üzerinden ulaşabileceği objeleri(*Item*) görebilir. Oyuncu, bu objelerin ne kadara satıldığını(Para), ne kadar zamanda bir kazanç sağlayacağını(Zaman), ne kadar deneyim(XP) ve para(Kira) kazanacağını ve son olarak ilgili objenin küçük bir resmini görebilir. Oyunun *use case* diyagramı şekil 3.4 de gösterilmiştir.

Sade bir tasarıma sahip olan oyunun, oyuncuyu yormaması hedeflenmiştir. Bu sayede oyuncu daha fazla zevk alacaktır. Oyun içerisindeki tarla ve bina gibi markette satılan objelerin tasarımı 3DS MAX ile diğer grafikler PHOTOSHOP yardımıyla yapılmıştır. Oyunda izometrik bir bakış açısı yakalamak oldukça önemlidir. Bunun için kod içerisinde kamera ortografik olarak ayarlanmıştır. Ayrıca izometrik bakış açısı simetrik objelerin 3 yüzeyini tam olarak gösterebildiği için tercih edilir.



Şekil 3.3 İzometrik Açı

İzometrik bakış açısı Şekil3.3'de görüldüğü üzere 45 dereceye 35.264 derecedir. 3DS MAX'de gerekli açı değerleri ayarlandıktan sonra rotasyon değerlerine tekrar bakılarak bu değerler oyun içerisinde kamera için ayarlanmıştır.

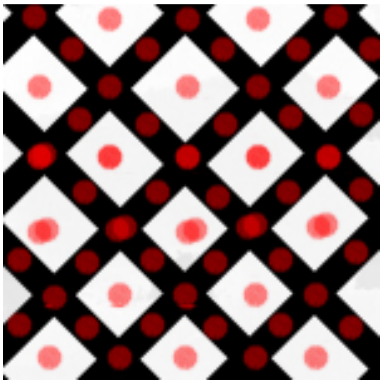


Şekil 3.4 Use case diyagramı

4. Halicville İstemcisine Genel Bakış

Objelerin zemine kitlenmesi, grafiksel değişimler, varsa animasyonların oynatılması gibi işlemleri dinamik olarak *Flash'ın* hesaplaması gerekir. Bunun yanı sıra oyuna üç boyutlu görünüm özelliği vermemizi sağlayan motor "Alternativa3D"dir. 3.parti bir motor olan Alternativa3D Adobe tarafından desteklenmektedir. Alternativa3D sayesinde 3dsmax programıyla yapılan üç boyutlu objeler oyuna entegre edilmiştir. Ayrıca bu objeler oyun içerisine yüklenirken aynı zamanda belirtilen klasörden ilgili kaplamaları çekerek objenin doğru bölümünü kaplamasını rahatlıkla sağlanabilir.

4.1.1. Objelerin Zemine Kitlenmesi

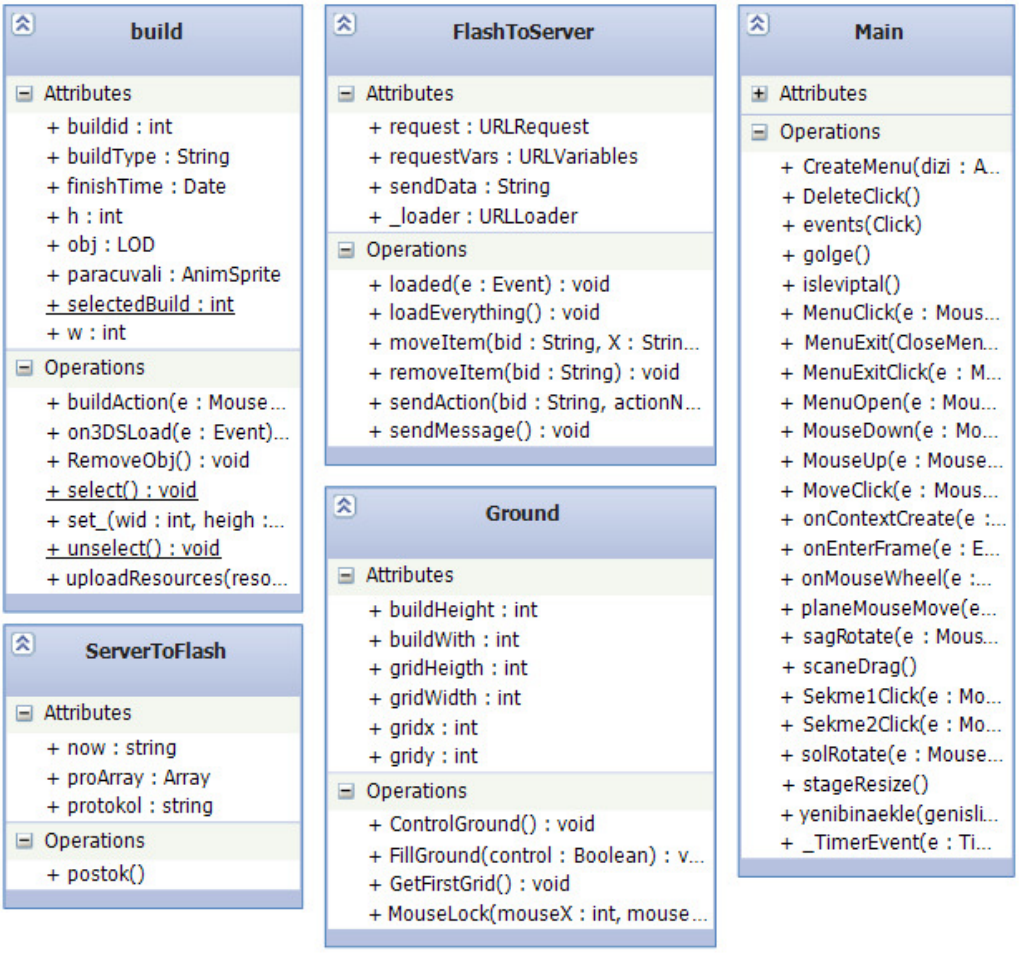


Şekil 4.1 Oyunun grid yapısı

Zemin dinamik olarak 20 piksele 20 piksellik karelere parçalanmıştır. Oyuna yüklenen üç boyutlu objeler Şekil 4.1'de görüldüğü üzere bu karelerin merkezine konumlanır. Bu kontrol "Ground" sınıfı ile yapılır. Eğer obje 40 piksele 40 piksel boyutundaysa zemin 20 piksel oranında bölündüğü için bu obje 2x2'lik grid alanı kaplamış olur.

4.2. Sınıflar

Flash içerisinde sınıflar ".as" uzantısıyla ifade edilirler. ".as" uzantılı dosya içerisindeki *package* özelliği atanmamış ise oluşturulan bu nesneler oyunun asıl dosyasının(".fla" uzantılı dosya) yanında bulunmalıdır.



Şekil 4.2 Sınıf diyagramları

4.2.1. Ground Sınıfı

Ground sınıfı içerisinde *MouseLock, FillGround, ControlGround ve GetFirstGrid,* adlı metotlar bulunmaktadır. Bu metotların yanı sıra *gridWidth, gridHeight, gridx, gridy, buildWith ve buildHeight* adlı değişkenler bulunmaktadır.

public const var gridWidth:int = 20,gridHeigth:int = 20;

var gridx:int,gridy:int;

public var buildWith:int,buildHeight:int;

Flash'ta değişkenler, (*public, private* vb.)(static)(*const, var* vb.)(değişkenin adı)(:)(değişkenin tipi)(=)(değişkene atanacak değer) şeklinde tanımlanmaktadır. *gridWith* ve *gridHeigth* adlı değişkenler *const* olarak tanımlanmıştır. Yani bu değişkenlere tek seferlik değer atanabilir. Daha önce 1x1'lik grid yapısının 20x20 pikselden oluşmaktadır. *gridWidth* ve *gridHeigth* değişkenleri bu değerleri taşımaktadır. Bu değerler oyun süresince değişmeyeceğinden dolayı bu değişkenler sabittir. *gridx* ve *gridy* değişkenlerinde objenin hangi grid üzerinde bulunduğu bilgisi saklanır. *buildWith* ve *buildHeight* adlı değişkenlerde ise objenin grid üzerinde kaça kaçlık alan kapladığı bilgisi tutulmaktadır. Flash'ta fonksiyon ya da diğer adıyla metot tanımlamak istediğimizde *function key word'unu* kullanılmaktadır.

public function MouseLock(mouseX:int,mouseY:int):Point

{

}

(public, private)(function)(fonksiyonun Adı)(fonksiyonun alacağı parametreler)(:)(Fonksiyonun geri dönüş tipi) şeklinde tanımlanmaktadır. *MouseLock* metodunda *mouse'un* ekran üzerindeki koordinatlarını alarak objeyi hangi gride yerleştireceğini hesaplar ve o gridin koordinatlarını geri döndürür. Burada karşılaşılan sorun ise 2x1'lik obje ile 1x1'lik objenin kitlenme algoritmasının farklı olmasıdır. Bu yüzden bu metod, *buildWith* ve *buildHeight* değişkenlerine bağlı olarak parçalara ayrılır.

FillGround ve *ControlGround* metodları birbirine çok benzemektedirler. Bir objeyi zemine sabitlemek istediğimizde orada başka bir objenin bulunup bulunmadığını kontrol etmek için *ControlGround* adlı metod kullanılır. *FillGround* ise objeyi zemine sabitlediğimizde o objenin zeminde işgal ettiği tüm gridleri işaretler. Örneğin 2x2'lik bir tarla için zeminde 4 adet grid ya da 4x2'lik bir ev için 8 adet grid işaretlenir. Böylece zemine 4x2'lik bir ev sabitlemek istenildiğinde *ControlGround* metodu 8 gridlik alanın boş olup olmadığını kontrol eder. Eğer bu alan boş ise *FillGround* metodu bu alanda artık bir objenin bulunduğunu belirtmek amacıyla bu alanı işaret-

ler. Bir objeyi taşımak istediğimizde ise eğer obje taşınabiliyorsa önceki konumundaki işaretleri kaldırır ve yeni konumu işaretler. Bu işaretler bu sınıfın dışında tanımlanmış olan *Boolean* tipindeki *groundControl* adlı dizide tutulur.

GetFirstGrid metodu *FillGround* ve *ControlGround* metodları içerisinde kullanılır. 4x2'lik ev için 8 adet gridin işaretlenir. Bu 8 adet grid işaretlenirken algoritmada hesaplama kolaylığı olması nedeniyle sol üst köşedeki gridin koordinatları hesaplanır. Eğer bu alanlar kontrol edilecekse sol üst köşedeki gridten itibaren, *buildWith* ve *buildHeight* değişkenlerine bağlı olarak döngü gerekli alanları kontrol eder. Bu değişkenler "boundBox" adı verilen sınıf içerisindeki "max" ve "min" değerleri kullanılarak bulunur. Eğer *FillGround* metodu ile bu alanların işaretlenmesi gerekiyorsa aynı şekilde sol üst köşeden itibaren döngüler yardımıyla gerekli koordinatlar işaretlenir.

4.2.2. Bina Sınıfı

Bina sınıfı içerisinde oyun içerisindeki objelere ait bilgiler tutulur. Oyuncunun oyun içerisinde oluşturduğu her obje bu sınıfın bir örneğidir. Bu sınıf içerisinde binaya özel bir id numarası, binanın tipini, binanın kaça kaçlık grid alanı kapladığını(Tarla için 2x2), o objeye ait süre bilgisini ve o objenin resim bilgisini barındırır. Tüm bu bilgiler için değişkenler tanımlanmıştır. Bu değişkenlerin adları sırasıyla *buildid, buildType, w, h, finishTime, obj ve paraCuvali'dır.* Bu değişkenlerin yanı sıra *static* olarak tanımlanmış *selectedbuild* değişkeni bulunmaktadır. Bu değişken oyun içerisindeki seçili olan objenin ID numarasını saklamaktadır.

Bina sınıfı içerisindeki tanımlanmış olan metodlar ise set_, uploadResources, on3DSLoad, unselect, select ve buildAction metodlarıdır.

- set_ metodu: Bina sınıfıyla ilgili olan tüm parametrelerin ilk değerlerini atamaya yarayan metotdur. Bu sınıfın bir örneği oluşturulduktan sonra eğer oyun ilk defa açılıyorsa server tarafından gelen aksi halde market sayfasından gelen bilgiler bu metod kullanılarak gerekli değişkenlere atanır. Ayrıca üç boyutlu objede burada yüklenir. Eğer o obje ilk defa yükleniyorsa ilgili klasörden çeker. Aksi halde sınıfın içerisinde o objenin kopyasını koyar. Bu sayede hafıza daha verimli kullanılmış olur.

- on3DSLoad metodu: Objelerin oyuna yüklenmesi tamamlanınca bu metod çalışır. Bu metod içerisinde ilgili "3ds" formatındaki dosyanın içerisindeki tüm üç boyutlu objeler "LOD" adlı sınıfın bir örneğine eklenir. Bu objelerin "LOD" adlı sınıfın örneğine eklenmesinin amacı daha önce bahsettiğimiz "boundBox" sınıfı içerisindeki değerlerin doğru hesaplanmasını sağlamak içindir. Aksi halde bu değerler sonsuz olarak görünecektir. Gene bu olay içerisinde üç boyutlu ilgili objeye ait tüm kaplamalar yüklenmektedir. Kaplamaların ve objenin ekranda görülebilmesi için "upload" işlemi gerçekleştirilir. Üç boyutlu obje ile ilgili "boundBox" hesaplaması "calculateboundBox" adlı metod ile yapılır. İlgili objeye ait bir olay gerçekleştiğinde örneğin bir evin kira süresi dolduğunda o obje üzerinde para çuvalı resmi görünür. Bunu yapabilmek için bir plane oluşturulduktan sonra kaplama bilgilerinde "alpha" tipindeki kaplamada eklenerek daha düzgün bir görünüm elde edilmiştir. Burada aksiyona bağlı olarak bu plane'in görünürlük özelliği değiştirilir.
- select metodu: Oyun içerisindeki bir objeyi seçili halde getirmek için kullanılır. Seçilen binaya ait bilgiler gerekli parametrelere atanır.
- unselect metodu: Seçili olan objenin seçimini iptal etmek için kullanılır. Hangi binanın seçili olduğunu söyleyen *selectedbuild* değişkenine -1 değeri atanır.
- buildAction metodu: Bir objeye tıklandığında kira toplama, tarlayı hasat etme ya da tarlayı temizlemek gibi işlemler bu metod yardımıyla yapılır. Binanın tipine göre gerekli if bloklarına bağlı olarak çalışır ve server'ı durumdan haberdar eder.
- uploadResources metodu: Yüklenen üç boyutlu objelerin ve kaplamaların ekranda update edilmesini sağlayan metotdur.

4.2.3. Roo Sınıfı

Roo sınıfı uygulamanın sahne(stage) kısmını hafızada tutmak amacıyla oluşturulmuştur. Uygulama ilk çalıştığında stage için yazılmış kodlar çalışır. Diğer sınıflar ancak burada örneklenirse ya da bir şekilde çağırılırsa çalışabilir. Diğer sınıflarda sahneye erişmek zahmetlidir. Bu yüzden bu sınıf içerisinde t adında bir değişken tutulur. Bu değişkene sahne atanır. Bu şekilde sahne üzerindeki tüm MovieClip nesnelerine, değişkenlere ya da fonksiyonlara erişilebilir.

4.2.4. MenuItem Sınıfı

Şekil 4.3 Menuitem movieclip görüntüsü

Market sayfasında her objeye ait bir takım bilgiler tutulur. Bu bilgiler ilgili objenin fiyatı, ne kadar zamanda bir gelir sağlayacağı, ne kadar kazanç getireceği gibi bilgilerdir. Her obje için bu bilgiler farklıdır. Bu yüzden bir *MenuItem* sınıfı tanımlanmıştır. Az önce belirtilen bilgiler bu sınıfın içerisinde bulunur. Market sayfası oluşturulacağı zaman ilgili sekmeye ait *MenuItem'lar* örneklenerek dizilir. Bu işlem tek seferlik gerçekleşir. Market sayfası kapatılmak istendiğinde görünürlük özelliklerine(visible property) *false* değeri atanır. Ayrıca bu sınıf içerisinde fare ile tıklanıp tıklanmadığını algılamak amacıyla bir olay(event) tanımlanmıştır. Bu *event* çalışıyorsa o *menuItem'a* ait obje, gerekli kontrollerden geçtikten sonra (oyuncunun parası yetiyor mu? gibi) oyuncunun satın alması üzere ekranda belirir. Oyuncunun faresine o obje kitlenir. Oyuncu bir yer belirleyip fareye tıkladığında objeyi yerleştirmiş olur.

4.2.5. FlashToServer Sınıfı

Bu sınıf ile server'a gerekli protokolleri göndermek amaçlanmıştır. Her protokol için ayrı ayrı metodlar oluşturulmuştur. Protokolleri gönderebilmek için URLLoader nesnesi örneklendi. URLRequest ile server'ın adresi tutuldu. Gönderilecek veriyi saklamak içinse URLVariables nesnesi örneklenmiştir. Böylelikle bu sınıfa ait tüm değişkenler tanımlanmıştır. URLLoader için ilgili adresin yenilenip yenilenmediğini anlamak amacıyla(postback) olay(event) oluşturulmuştur. Tüm bu işlemlerden sonra verileri göndermek çok kolaylaşmıştır. URLRequest'ın data property'sine URLVariables nesnesine ait değişken atanır. URLVariables nesnesine ait değişkene gönderilmek istenen değişken atanarak, veriler URLLoader nesnesi yardımıyla gönderilir. Bu işlemi ise URLLoader nesnesinde tanımlanmış olan load metodu ile yapılmıştır.

4.2.6. ServerToFlash Sınıfı

Bu sınıf yardımıyla server'dan gelen bilgiler dinlenir. Sever bir protokol gönderdiğinde bu sınıf çalışır. Protokolleri parçalamak ve içerisindeki verilere ulaşmak amacıyla *postok* adlı metod tanımlanmıştır. String tipinde Data adlı bir parametre almaktadır. Bu parametrede server'ın gönderdiği prokoller bulunur. Protokoller bu metod içerisinde parçalanarak, market bilgisi, bir objenin silindiği bilgisi, oyuncunun para ve puan bilgisi ya da objelere ait detaylı bilgiler elde edilir.

4.2.7. Ana Flash

Market sayfasında *menuItem* nesnelerini saklamak için bir dizi tutulmuştur. Bu dizinin amacı market sayfası kapatıldığında bu nesnelerin visible özelliğini false olarak, market sayfası açıldığında ise true olarak değiştirmektir. Ayrıca yeni bir obje almak istediğimizde o objenin satın alınıp alınamayacağına bu dizideki ilgili objenin fiyat bilgisiyle oyuncunun fiyat bilgisini karşılaştırarak yapılabilir. Benzer şekilde diğer bilgiler de kullanabilir. *genislik* ve *yükseklik* değişkenleri *menuItem'ları* market sayfasına yerleştirirken koordinat hesabı yapmak için kullanılırlar. *openedSekme* değişkeni en son açık kalan sekme bilgisini saklar.

Market sayfasında sekmeler için ve market sayfasını kapatmak için olaylar(event) yazılmıştır. Bir sekmeye fare ile tıklandığında eski sekmeyi *MenuExit* metodu ile kapatır. Yeni sekmeyi ise *MenuOpen* metodu ile açar. Market sayfasını tamamen kapatmak için sekmelerde olduğu gibi *MenuExit* metodu kullanılır. *MenuExit* metodu boolean tipinde parametre almaktadır. Bu parametre true olarak girildiğinde Market sayfası tamamen kapatılır. False değeri girildiğindeyse sadece ilgili sekme kapatılır. *MenuExit* ve *MenuOpen* metodlarında, market sayfasına ait bilgilerin açılıp kapatılması, *menuItem* dizisinde döngüler yardımıyla içerisinde dolaşarak, visible özelliğini değiştirmemizle oluşur.

CreateMenu metodu ise market sayfasına ait bilgilerin ilk defa oluşturulması sırasında çağırılır. Bu metod içerisinde ilgili objelere ait değerlerin ilk değerleri(initial value) atanır. *genişlik* ve *yükseklik* değişkenleri bu metod içerisinde kullanılarak *menuItem'ların* koordinatları belirlenir.

Dinamik olarak zemine eklenen objeler(tarla ya da ev gibi) henüz bu görsellerden ayrı değildir. Bu ayrımı sağlayabilmek için algoritma içerisinde *rootContainer* adlı

Object3D nesnesi tanımlanmıştır. Ekleyeceğimiz objeleri bu *Object3D* nesnesi içerisine ekleyerek objeleri diğer görsellerden ayrılmıştır.

Boş bir tarlaya ürün ekmek istendiğinde önce ekilmek istenen ürün seçilmelidir. Daha sonra tarlayı seçerek ilgili ürün ilgili tarlaya atanabilir. Burada seçilen ürünü saklayan değişken *selectedFarm* ile ifade edilmiştir. Bir objede bir aksiyonun oluşup oluşmadığına yönelik kontrol yapılmasında yardımcı olan değişken *actionControl'dur*. Objelerin zemine kitlenmesi için Ground sınıfı türetilmiştir. t*ransport* ve *moveControl* adlı değişkenler bir obje taşınmak istendiğinde set edilirler. *removeControl* adlı değişken bir objeyi silmek istendiğinde set edilir. Objeler için binalar adlı bir dizi oluşturulmuştur. Server'a bilgi göndermek için FlashToServer nesnesi türetilmiştir. Son olarak aksiyonların düzenli çalışabilmesi için 1000 milisaniye intervale sahip bir timer oluşturulmuştur. Kamerayı diğer objelerden ayırmak için cameraContainer adlı Object3D nesnesi oluşturulmuştur. Camera3D tipinde kamera nesnesi tanımlamış ve cameraContainer içerisine eklenmiştir. Objelerin konulabileceği alanı sınırlandırmak açısından bir *plane* tipinde zemin oluşturulmuştur. Kameranın sağa, sola, yukarı ve aşağı kaymasını sağlamak amacıyla SimpleObjectController nesnesi tanımlanmıştır.

Sahne içerisinde tanımlanan metodlar ise, constructor metodu, events, yenibinaekle, SetObjectXY, isleviptal, _TimerEvent, MenuClick, MoveClick, DeleteClick, onEnterFrame, MouseDownEvent, MouseUpEvent, stageResize gibi metodlardır.

- constructor metodu: Flash ilk çalıştığında bu metodu çalıştırır. Bu metod içerisinde sahne bilgisi "Roo" sınıfı içerisindeki "t" değişkenine atanır. Flash'ın tüm bilgileri çekebilmesi için "FlashToServer" sınıfı içerisindeki "loadEverything" metodu çağırılır. Sahne sol üst köşeye hizalanır ve genişleme modu kapatılır. Kamera nesnesinin örneği oluşturulur. Sahneye eklenir. İzometrik açıda olacak şekilde rotasyonları ayarlanır. Ve orthographic ayarı açılır. Ekran üzerinde görünecek market butonu, taşıma ve silme butonu gibi movieClip nesneleri ekrana eklenir. Ve ReSize metodu kullanılarak ekran üzerinde konumlandırılır. Olayların tanımlandığı metod çağırılır.
- events metodu: Sahne üzerinde tanımlanmış tüm olaylar(events) bu metod içerisinde bulunur. Bu metod içerisinde "timer" başlatılmıştır. Butonların tıklanma(click) olayları tanımlanmıştır. Mouse için tuşa basıldığı ve bırakıldığı

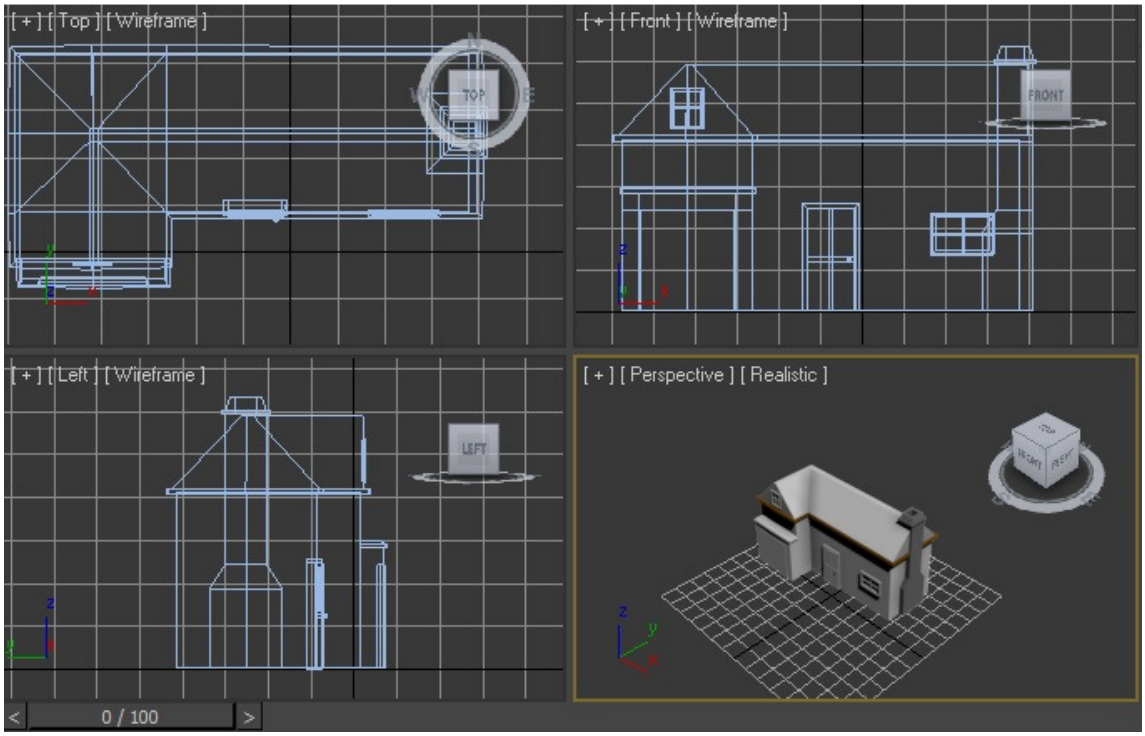
bilgisini veren olaylar tanımlanmıştır. Ekranın boyunun değişmesi ihtimaline karşılık ekran üzerindeki butonların ve bilgi ekranının konumlarının değişmemesi içinde bir olay tanımlanmıştır.

- yenibinaekle metodu: Bu metod içerisinde "build" sınıfında tanımlanmış olan "set_" metodu kullanılarak yeni obje eklenmesi amaçlanır. Öncelikle binalar dizisine kaydedilen obje için gerekli bilgiler set edilir. Oyuncunun objeyi istediği yere sabitleyebilmesi için "moveControl" değişkenine "true" değeri atanır.
- SetObjectXY metodu: Bir objeyi taşımak istediğimizde objenin yeni koordinatlarını set etmek için bu metodu kullanırız.
- isleviptal metodu: Bir butona tıkladıktan sonra(objeyi taşımak ya da silmek için) gerekli işlemi uygulamadan başka bir butona tıklarsak oyun içerisinde hatalar oluşur. Bu yüzden eğer bir objeyi taşımak üzere "moveControl" değişkenini set etmişsek ancak bir objeyi taşımadan karar değiştirip bir objeyi silmek istediğimizde "moveControl" değişkenini "false" değerine dönüştürüp sadece "removeControl" değişkenine gerekli atamayı yapmalıyız. Bu nedenle bu metod içerisinde sorun çıkarabilecek bu tarz değişkenlere "false" değeri atanır.
- TimerEvent metodu: Binalar dizisine bakılarak bir tarlanın hasat vaktinin gelip gelmediğini ya da bir evin kira süresinin dolup dolmadığını sürekli olarak kontrol edilmelidir. Bu yüzden bir saniye arayla çalışan bu metod tanımlanmıştır. Bir diğer sorun ise bir obje taşınmak istendiğinde başka objelerin üstüne çıkmasıdır. Bu yüzden objelerin "setChildIndex" değerleri atanarak hangi objenin daha yukarda görüneceğini ayarlanır.
- MenuClick metodu: Market sayfasına ulaşmak için tanımlanmış bir metottur. Burada "isleviptal" ve "MenuOpen" metodları çağırılır.
- MoveClick metodu: Bir objeyi taşımak üzere etkileşimde bulunduğumuzda bu metod çalışır.
- DeleteClick metodu: Bir objeyi taşımak üzere etkileşimde bulunduğumuzda bu metod çalışır.
- onEnterFrame metodu: Bu metod, uygulama çalışmaya başladığı ilk andan itibaren çalışabileceği en hızlı şekilde sürekli olarak çalışır. Kameranın sürekli olarak render alması burada sağlanır.

- MouseDownEvent metodu: Mouse'a basılı tutuğumuzda zemini kaydırmamızı sağlamak için "startdrag" metodu kullanılmıştır. Mouse'daki kayma miktarını öğrenebilmek nedeniyle mouse'un koordinatları kaydedilir. "MouseUpEvent" metodunda gerekli kontrolleri yapmak için "actionControl" değişkeni başlangıç değeri olarak "false" değerine set edilir.
- MouseUpEvent metodu: Zemin'in sonsuza kadar kaymasını önlemek amacıyla zemin için "stopDrag" metodu burada tanımlanmıştır. Ayrıca mouse'un yeni koordinat bilgileri elde edilerek "MouseDownEvent" metodundaki değerlerden çıkartılır. Böylelikle mouse'un ne kadar yer değiştirdiği tespit edilir. Eğer belli bir sınırın altındaysa gerekli işlemler yapılabilir. Aksi halde sadece zemini kaydırmaya ve durdurmaya yönelik işlemler yapılır. Eğer taşıma ya da silme gibi işlemler yapılmıyorsa sadece objeye tıklanmışsa bu durumda bir aksiyon oluşması gerekir. Bu nedenle "actionControl" değişkeni "true" set edilir. Aksi halde "false" olarak kalır. Eğer bina zemine yerleştirilebiliyorsa yerleştirilir, yerleştirilemiyorsa hata mesajı verir. Son olarak gerçekten objeye mi tıklandı yoksa transparan alana mı tıklandığı kontrolünü yapmamıza yardımcı olan "HitTester" içerisindeki "realHitTest" metodu burada kullanılır.
- stageResize metodu: Eğer sahne büyümüşse ya da küçülmüşse sahne üzerindeki butonların ya da para ve xp alanını gösteren görselin konumlarının korunması nedeniyle bu metoda ihtiyaç duyulmaktadır. Bu metod sadece sahnenin boyu değiştiğinde çalışır.
- scaneDrag metodu: Bu metod içerisinde kameranın hangi durumlarda nasıl hareket edeceği tanımlanmıştır.
- onMouseWeel: Mouse'un üzerindeki topun ileri yada geri gitmesiyle bu metod çalışır. Burada kamera için tanımlanmış olan cameraContainer objesinin büyüklüğü değiştirilerek kameranın sahneye yakınlaşıp uzaklaşması sağlanır.

5. 3DS MAX ve PHOTOSHOP ile GÖRSELLERİN HAZIRLANMASI

5.1. 3DS Max Yazılımı



Şekil 5.1: 3D Studio Max içerisinden bir görüntü

Objelerin izometrik olarak mükemmel şekilde çizilmesini sağlamak için 3DS MAX programı kullanılmıştır. Objelerin çizimini yapabilmek için basit bir "box" objesi çizilmiştir. Bu objeyi değiştirebilmek(editable) için objeye sağ tıklayarak ya da "modifier" menüsünden "Editable Mesh" ya da "EditablePoly" seçeneklerinden biri kullanılır. Ancak "EditablePoly" modifier'inde daha fazla seçenek bulunduğu için bu modifier(değiştirici) seçilmiştir. Bu modifier sayesinde objenin en alt katmanları olan vertex(nokta), edge(kenar), poygon(dörtgen) ve element(objenin tamamı) tipinde

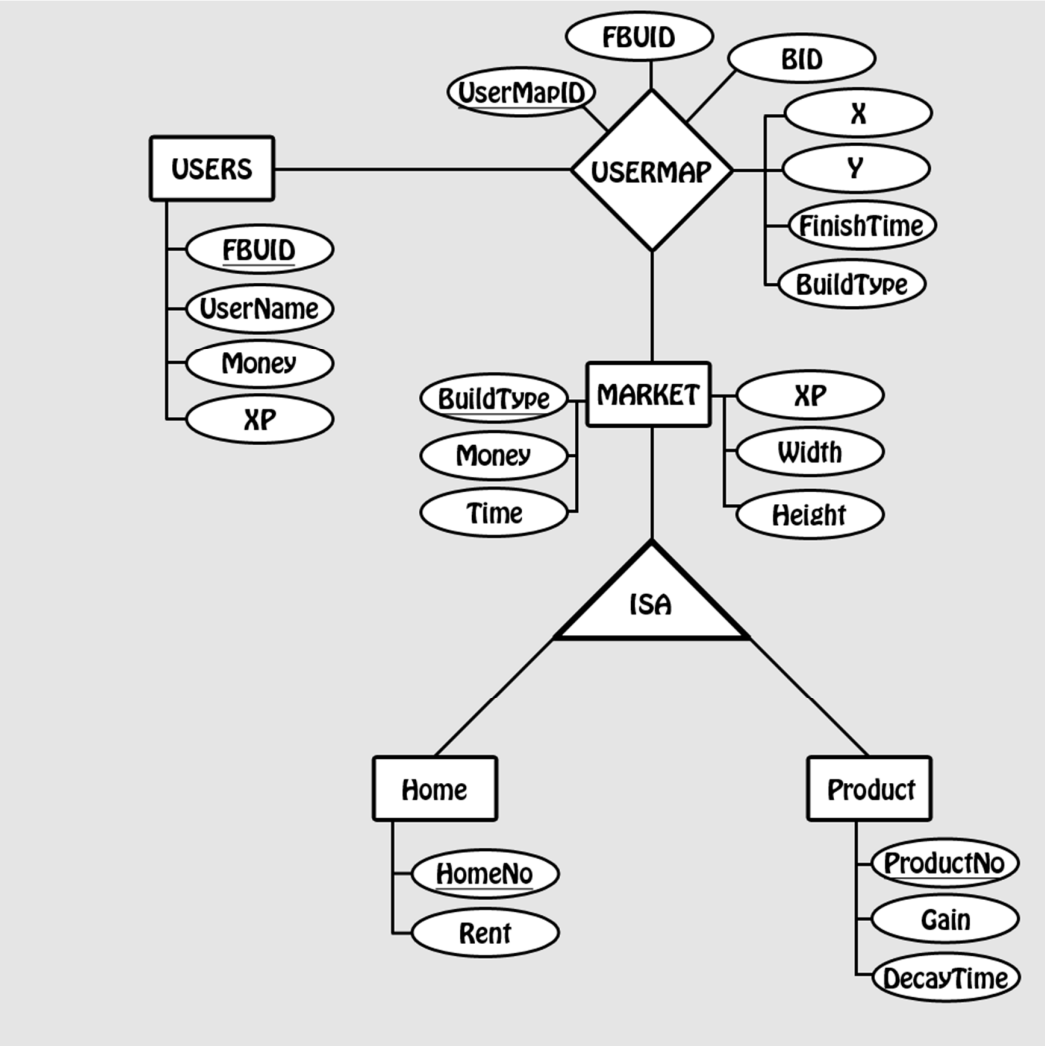
değişiklik yapılabilmektedir. Obje ev şekline yakın bir hale getirildiğinde objenin eve benzemesini sağlamak amacıyla obje üzerine kaplamalar atanır. Bu işlemi yapabilmek için "M" kısayol tuşu kullanılarak "Material Editör" 'e ulaşılır. 3DS Max için yazılmış özel hesaplama algoritmalarına sahip vray motoru kullanılarak daha güzel sonuçlar alınabileceğinden kaplamalarda da bu engine'e özel kaplamalar kullanılmıştır. Vray Material kaplamasına resimler atanarak obje kaplanır. Resim ile objenin düzgün bir şekilde kaplanmasını sağlamak amacıyla objeye "UVW MAP" modifier'i atanır. Bu şekilde resmin koordinatlarıyla objenin koordinatları eşlenir. Modelleme ve kaplama gibi işlemlerden sonra kamera, ışık ve render ayarlarına geçilebilir. Kaplamaların render sonucunda farklı çıkmasını önlemek amacıyla öncelikle ışık ayarı da yapılabilir. Kamera izometrik açılara getirilerek objeye uygun bir şekilde yerleştirilir. Işıklarda ayarlandıktan sonra sonucu görmek üzere "Render" tuşuna basılır.

5.2. Photoshop Yazılımı

Photoshop üzerinde katman yapılarıyla çalışmanın büyük önemi vardır. Bir değişiklik yapılacağı zaman sadece ilgili katman üzerinde değişiklik yaparak diğer bölgeler etkilenmeden çalışmak mümkündür. Katmanlara hızlı bir şekilde stiller atanabilir ya da bir katmana çift tıklayarak yeni bir stil oluşturulabilir. *Graphic Tablet* kullanarak bir kağıda çizim yapıyormuş gibi rahat bir şekilde çalışılabilir. Katmanların Hue/Saturation değerlerini değiştirerek aynı görseli farklı tonlara kavuşturulabilir. "*Clone Stamp Tool*" ile resmin bir noktasını farklı bir bölgeye klonlayarak değişik görseller elde edilebilir. Son olarak vektörler yardımıyla çok fazla yetenek istemeyen işlemlerde yapılabilir. Bir katman üzerinde vektörel maskeleme atanarak görseli istediğimiz şekilde sınırlandırılabilir. Halicville uygulamasında genel anlamda bu yöntem kullanılmıştır. Vektörlerle çalışılarak ve bir takım hazır stiller kullanılarak görseller hazırlanmıştır.

6. VERİ TABANI

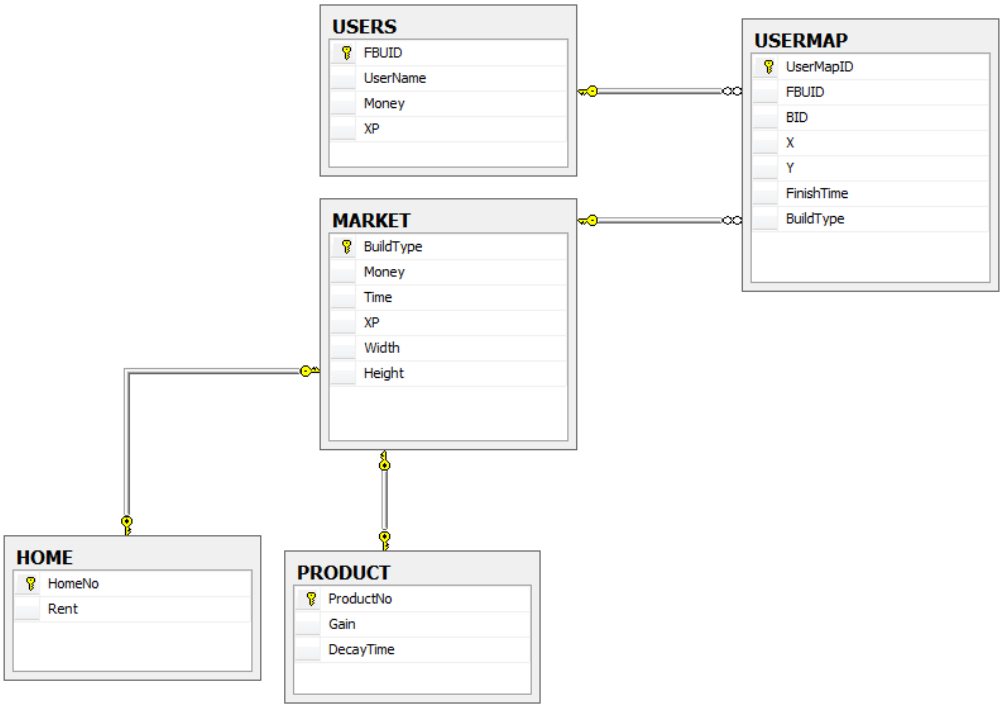
6.1. E/R Diyagramı



Şekil 6.1: E/R Diyagramı

Oyunun veri tabanı şekildeki gibidir. Oyuncular ya da kullanıcılar için oluşturulmuş "USERS" tablosunda birincil anahtar(primary key), facebook user ID olarak tanım-

lanmıştır. Her kullanıcının facebook kullanıcı ID'sı birbirinden farklıdır. Ayrıca bu tabloda oyuncuların kullanıcı adları, sahip oldukları oyun parası ve XP miktarı da saklanmaktadır. USERS tablosu ile MARKET tablosu arasında USERMAP ilişkisi kurulmuştur. Bu sayede oyuncunun marketten aldığı dolayısıyla haritasında bulunduğu tüm objeler USERMAP tablosu içerisinde tutulur. USERMAP tablosu birincil anahtar olarak UserMapID'yi kullanır. FBUID ve BuildType üzerinden USERS ve MARKET tablolarıyla bağlantı kurar. USERMAP tablosu, oyuncuların facebook ID'lerini, Flash içerisinde objeler için tanımlanmış ID numaraları, objelerin x ve y koordinatları, objelerin varsa bitiş süreleri(kiranın toplanma zamanı, tarlanın hasat edilme zamanı gibi) ve son olarak objelerin tiplerini saklar.



Şekil 6.2: SQL Server içinden diyagram yapısı

"MARKET" tablosu oyun içerisinde satın alınabilecek tüm objeleri barındırır. Dolayısıyla bu tablo içerisinde objelerin tiplerini(BuildType), ne kadara satın alınabileceklerini(Money), aksiyon zamanlarını(Time), oyuncuya aksiyon sonunda ne kadar bonus kazandıracağı(XP) ve objenin grid üzerinde kaça kaçlık alan kapladığı(width ve height) bilgileri saklanır. Ayrıca objelerin az önce saydığımız ortak verilerin yanı

sıra kendilerine ait verileri de bulunduğu için ISA bağıntısı kullanılarak MARKET tablosundan ayrılmışlardır. HOME tablosunda binaların ne kadar kira vereceğini tutmak için *Rent* sütunu tanımlanmıştır. PRODUCT tablosunda ise hasat edilen ürünün ne kadar kazanç sağlayacağı ve ürün toplanmazsa eğer ne kadar süre sonra bozulacağı *Gain* ve *DecayTime* ile tutulmaktadır. HOME ve PRODUCT tabloları HomeNo ve ProductNo birincil anahtarları üzerinden MARKET tablosuna BuildType sütununa bağlanmışlardır.

6.2. Saklı Yordam

Saklı yordamlar SQL Server içerisinde derlenmiş olarak bulunacaklarından daha hızlı çalışırlar. GetMarketHome ve GetMarketProduct ile tüm market sayfası veri tabanından çekilebilir. GetUser ile oyuncu oyunu başlatacağı zaman para ve bonus bilgisini çekebilmek için kullanılır. UpdateUser ile oyuncu oyun sırasında marketten aldığı objelerle ya da elde ettiği kazançlarla para ve bonus bilgisini güncellerken kullanılır. UsermapInsertWithFBUID ile kullanıcı haritasına bir obje eklediğinde veri tabanında gerekli güncellemeleri yapmak için bu saklı yordamı kullanılır. UsermapRemoveWithBID ile oyuncu haritasından bir obje silmek istediğinde bu saklı yordamı kullanılır. Oyuncu bir objenin yerini değiştirdiğinde UsermapUpdateWithBID kullanılır. Oyuncu oyunu çalıştırdığında oyuna dair ilk bilgilerin yüklenmesini sağlamak için UsermapWithFBUID'yı çağırılır.

7. ASP.NET

Flash tarafından gönderilen protokolleri işleyerek doğruluğunu kontrol edip veri tabanında gerekli işlemleri yapan taraftır. Bu şekilde oyuncunun hile yapması engellenir. Çünkü oyuncunun her hareketi değerlendirilerek veri tabanında değişiklikler yapılır. Bu nedenle oyuncu hesabına bir anda yüklü miktarda para yükleyemez. Yapabileceği tek eylem oyunu kendi kendine oynayabilecek bir bot yazarak oyuna müdahale etmek olabilir.

7.1. Sınıflar

7.1.1. Ground Sınıfı

Flash içerisinde yapılan kontrollerin aynısı yapılmaktadır. Zemin'e bir obje eklendiğinde ya da silindiğinde FillGround metodu çalışır. Zemine bir obje ekleneceği zaman başka bir objenin var olup olmadığını kontrol etmek amacıyla ControlGround metodu kullanılır. GetFirstGrid Flash'takine benzer şekilde sol üst köşedeki gridi yakalamak için kullanılır. GoLeft ve GoUp metodları gridler arasında dolaşabilmek için gereklidir.

7.1.2. SQLProcess Sınıfı

Three Tier Architecture diğer adıyla üç katmanlı yapıyı oluşturmak amacıyla bu sınıf tanımlanmıştır. Bilindiği üzere üç katmanlı yapı Presentation Layer, Business Logic Layer ve Data Access Layer'dan oluşur. Burada Data Access Layer yapısını SQLProcess ile sağlamaktayız. Diğer katmanlar protokol sayfasına gömülmüştür.

Bu sınıf içerisinde SQL Server'a bağlanırken kullandığımız bağlantı metni yani connection string, ConStr değişkeninde tanımlanmıştır. SQL Server'a bağlanmamızı sağlayan SQLConnection sınıfı private olarak tanımlanmıştır. Connect metodu içerisinde SQLConnection, ConStr kullanılarak örneklenmiştir. Ve veri tabanına bağlantıyı açmak için Open metodu çağırılmıştır. Benzer şekilde kendi tanımladığımız Close metodunda SQLConnection içerisideki Close metodu çağırılarak bağlantının kapatılması amaçlanmıştır.

Herhangi bir parametre almadan direk çalıştırılacak saklı yordamlar ve parametre alarak çalışan saklı yordamlar bulunmaktadır. SQLProcess sınıfı static bir sınıf olduğu için overload yapılamamaktadır. Bu nedenle dört farklı metod tanımlanmıştır. Bu metodlara kısaca bakmak gerekirse;

- ListStoredProcedureWithoutParam: Parametre tanımlamadan çalıştırabileceğimiz metod tipidir. Bu metod sonucunda veri tabanındaki veriler Datatable nesnesine aktarılarak return edilir.
- ExecuteStoredProcedureWithoutParam: Parametre tanımlamadan çalıştırabileceğimiz metod tipidir. Bu metodu INSERT, UPDATE ve DELETE işlemleri için kullanırız. Bu nedenle geriye tablo verileri dönmez. İşlemin başarılı ya da başarısız oluşuna göre 0(hiçbir veri değişmedi), +1(bir veri değişti) ya da -1(sorgunun çalıştırılması sırasında bir hata oluştu) değeri geri döner.
- ExecuteStoredProcedure: Parametre ile çalışır. INSERT, UPDATE ve DELETE işlemleri için kullanılır.
- ListStoredProcedure: Parametre ile çalışır. SELECT işlemi için kullanılır.

Bu metodlar içerisinde SqlCommand kullanılır. Bu sayede saklı yordamın adı ve bağlantı tipi SqlCommand içerisinde belirtilmiş olur. Eğer gerekiyorsa SqlCommand içerisinde parametreler tanımlanır. Stored Procedure kullanıldığını belirtmek amacıyla SqlCommand sınıfındaki CommandType değişkenine CommandType.StoredProcedure değeri atanır.

Eğer geriye tablo verileri dönecekse SqlDataAdapter kullanılarak Datatable doldurulur. Aksi halde ExecuteNonQuery metodu kullanılır.

7.2. Protokol Sayfası

7.2.1. Değişkenler

Öncelikli olarak birkaç tane global değişkenler tanımlanmıştır. Flash içerisindeki Ground sınıfı burada da tanımlanmıştır. Bu değişkenler tıpkı Flash tarafında olduğu gibi zeminle ilgili kontrolleri yapabilmek adına bir Ground sınıfı içerisindeki groundControl hashtable'ı tanımlanmıştır. Bu dizi zemine yerleştirilen objelerin zeminde kapladıkları alanları boolean olarak hafızasında tutulmaktadır. Bu şekilde Flash tarafında olduğu gibi zemine yeni bir obje ekleneceği zaman ya da var olan bir obje-

nin yeri değiştirileceği zaman bu dizi üzerinden işlemler yapılmaktadır. Ve groundControl adlı hashtable ile ilgili olan işlemler gene bu sınıf içerisinde yürütülmektedir. Bu yüzden protokol sınıfına özel, global olarak Ground sınıfı örneklenmiştir.

Protokol sayfası bilgileri aldıkça yenilenmektedir. Bu yüzden birtakım önemli değişkenler kaybolmaktadır. Bu değişkenlerin kaybolmaması için Application Object ve Session Object kullanılmaktadır. Application Object tüm oturum bilgisini hafızasında tuttuğu için kişiye özel bilgiler burada tutulmaz. Genel bilgileri ve tek seferliğine kaydedilerek sürekli olarak kullanabilecek bilgileri tutmak daha faydalı olacaktır. Bu yüzden market sayfasını application object içerisinde saklarız. Bunun nedeni her obje işleminde sürekli olarak veri tabanına bağlanmayı önlemek, bu verilerin hiç değişmiyor olması ve sürekli olarak kullanılıyor olmasıdır. Bu durumda az önce bahsettiğimiz widthHeight hashtable'ını application object kullanarak saklamamız mantıklı olacaktır. Ground sınıfı içerisindeki groundControl hashtable'ını ise session kullanarak saklarız. Bunun nedeni ise bu bilgiler kişiye özeldir. Herkes için ayrı ayrı objelerin zeminde kapladığı alan bilgisi tutulur. Kullanıcı oyundan ayrıldıktan sonra bu bilgiler hafızadan silinir. Oyunu tekrar başlattığında veri tabanından(USERMAP) çekilen bilgilere(x ve y sütunları) bağlı olarak doldurulur.

Bu sayfada event olarak sadece Page_Load olayı çalışmaktadır. Sayfa yenilendiğinde yeni protokoller gelmiş demektir. Ve gerekli işlemler yapılarak sayfaya, Flash'a gönderilecek protokollerin yazdırılması için Response.Write metodu kullanılır. Bu metod ile sayfaya bilgiler yazdırıldığında bu bilgiler Flash tarafından Request edilerek alınmaktadır. Ve çevrim bu şekilde devam etmektedir.

Page_Load olayı içerisinde öncelikli olarak application ve session nesneleri ilgili değişkenlere eşitlenmiştir. Bu değişkenler daha önceden de belirtildiği gibi hashtable olarak kullanılmaktadır. Bu sayede key ve value değerleri ile diziye oranla daha rahat bir kullanım sağlanmıştır. widthHeight hashtable'ı key olarak buildType değerini almaktadır. Dolayısıyla aranan bilgilere sıklıkla kullanılan buildType değişkeni üzerinden erişilmiş olur. groundControl hashtable'ına ise objelerin x ve y koordinatları üzerinden(x.toString() + "x" + y.toString()) erişilmektedir.

7.2.2. Metodlar ve Eventler

Bu sayfada sayfa yüklenirken çağırılan Page_Load event'i, Market sayfası bilgilerini Flash'a göndermek için kullanılan MenuProtokol, Bina tipi dönüşümleri için BuildTypeToString ve BuildTypeToInt, uzun zaman bilgisinden kalan zamanı hesaplayan TimeSpanToSecond metodları kullanılmıştır.

- Page_Load: Öncelikle Application nesnesinin dolu olup olmadığı kontrol edilir. Eğer dolu ise bilgiler widthHeight Hashtable'ına atanır. Session objesi benzer şekilde kontrol edilir. Flash tarafından gelen bilgileri atamak için data değişkeni oluşturulur. Protokolün yapısına göre parçalanır. Parçalama işleminin ardından elde edilen veriler değerlendirilerek gerekli işlemler yapılır. Eğer parçalanan ilk veri "A" ise Action oluşmuş demektir. Bu durumda tarlayı hasat etmek üzere ya da kira toplamak üzere kontroller yapılır. Veri tabanından binaid'sine göre bina tipine ulaşılır. Bina tipinden widthHeight dizisi yardımıyla binanın özelliklerine(parası, ne kadar bonus getirdiği vs.) ulaşılır. Bina tipine göre kod ikiye(Ev yada tarla için) ayrılır. Eğer ev için kira toplanacaksa zaman kontrol edilir. Eğer sorun yoksa veri tabanında o binaya ait bilgiler (zaman) güncellenir. Daha sonra oyuncunun para ve bonus bilgisi de güncellendikten sonra Protokoller Flash'a gönderilir. Eğer bina tipi tarla olarak gelmişse öncelikle normal zaman(çürümemiş hali) kontrol edilir. Eğer bu testten geçerse çürüme zamanı da eklenerek çürüyüp çürümediği kontrol edilir. Tarla ve zaman bilgisi veri tabanında güncellenir. Eğer tarla çürümemişse para bilgisi de güncellenerek gerekli protokoller yollanır. Son olarak tarlaya ürün ekmek istediğine dair bir bilgi gelmişse Flash'tan gelen bilginin boş tarla bilgisi olup olmadığı kontrol edilir. Daha sonra oyuncunun parasının yetip yetmediği kontrol edilir. Eğer parası yetiyorsa veri tabanındaki para bilgisi güncellenir. Bina tipi veri tabanında oyuncunun ekmeye çalıştığı tarla olarak ayarlanır. Ve gerekli protokoller gönderilir. Eğer ilk veri "T" olarak gelmişse bir objenin yeri değiştirilmek isteniyordur. Protokol içerisinde ayrıştırılan bina tipi ile binanın özellikleri bulunur. Ground sınıfı içerisindeki binanın genişlik ve yükseklik bilgisi atanır. Veri tabanından o binaid'ye sahip objenin olup olmadığı test edilir. Eğer varsa binanın yeni koordinatları veri tabanında düzeltilir. Eğer o binaid'ye sahip bina yoksa yeni bina kuruluyor demektir. Bu durumda oyuncunun para bilgisi kontrol edilerek veri tabanında insert iş-

lemi yapılır. Flash'a protokoller gönderilir. Eğer ilk parçada "E" bilgisi geliyorsa obje silinmek isteniyordur. Bu durumda protokol içerisinde gelen binaid ile veri tabanında delete işlemi yapılır. Flash'a ilgili protokol gönderilir. Eğer ilk parçada "M" bilgisi gelmişse veri tabanından tüm market bilgisi çekilerek protokollere gömülür ve Flash'a gönderilir. Eğer ilk parçada "0" bilgisi gelmişse oyuncunun harita bilgisi Flash'a gönderilir. Veri tabanından USERMAP tablosu içerisindeki veriler facebook numarasına göre çekilerek Flash'a gönderilir.

- MenuProtokol: Market ile ilgili veri tabanından çekilen bilgiler protokollere dönüştürülerek Flash'a gönderilir. Bu sırada widthHeight dizisi eğer boş ise doldurulur.
- BuildTypeToString: Veri tabanından çekilen 10000, 10001 gibi rakamlar protokolde çok fazla yer kaplayacağı için bu değerler bu metodlar sayesinde e1, e2 gibi değerlere dönüştürülür.
- BuildTypeToInt: Benzer şekilde BuildTypeToString metodunda olduğu gibi e1, e2 gibi değerler 10000 gibi değerlere dönüştürülür.
- TimeSpanToSecond: Uzun zaman bilgisi olarak alınan değerler protokollerde çok fazla yer kapladığı için kalan zaman(saniye olarak) hesaplanır. Kalan zaman verisi Flash'a gönderilir. Bunu yapabilmek için TimeSpan sınıfı kullanılır.

7.3. Default Sayfası

PHP'den get yöntemiyle alınan facebook kullanıcı numarası veri tabanında aranır eğer yoksa oyuncu veri tabanına kaydedilir ve oyun başlatılır. Eğer oyuncu veri tabanında zaten kayıtlı ise oyun direk başlar.

PHP'den alınan bilgilerin doğruluğunu ise hash fonksiyonu ile kontrol ederiz. Facebook numarası|TEZHASH|yıl-ay-gün|saat olarak protokol md5 ile hash'lenir. Bu hash ve facebook numarası get ile Default.aspx'e gelir. Burada facebook numarası ile az önceki protokol tekrar hash'lenerek iki hash stringi karşılaştırılır. Eğer birbirinin aynısı ise oyun başlatılır. Zaman sorunu oluşmaması nedeniyle birkaç saat geriye giderek de kontrol yapılır.

8. PROTOKOLLER

Protokoller birbirinden "|~|" karakterleriyle ayrılırlar. Server ile Flash arasındaki protokoller daha kapsamlı incelenecek olursa.

- Market Protokolü: Flash açıldığında tüm market bilgisini çektiği için, Server'a yanlızca "M" karakterini gönderir. Server bu duruma karşılık, "M|bina tipi|Para|Kaç Saatte|Kaç birim|Genişlik|Yükseklik|Değişken" protokolünü gönderir. Yani objenin tipini, ne kadara satıldığını, kaç saniyede bir ürünün hasat edilebileceği ya da kira toplanabileceği bilgisi, sürenin sonunda ne kadar XP ya da bonus kazanılacağı, objenin genişlik ve yükseklik bilgisi ve son olarak veri tabanında da bahsettiğimiz gibi ISA bağıntısıyla ayırılan objelerin kendilerine ait özellikleri bulunmaktadır.
- Objelerin silinmesi: "E|bina id" bilgisi Flash tarafından Server'a gönderilir. Bu protokol Server'a o bina id'ye sahip objenin silineceğini söylemektedir. Bu duruma karşılık Server Flash'a "E|bina id|1 ya da 0 bilgisini gönderir. Burada protokolün sonundaki 1 o objenin silindiğini 0 ise bir hata meydana geldiğini belirtmektedir.
- Objelerin taşınması: Flash "T|Bina id|X|Y" ya da "T|Bina id|X|Y|Bina Tipi" protokollerinden birini server'a göndermektedir. İlk protokol var olan binanın yerini değiştirmek için kullanılır. Bu protokolde bina tipine gerek yoktur. Gelen X ve Y koordinatları mantıklı ise(bu koordinatlarda başka bir obje yoksa) o objenin koordinatları güncellenir. İkinci protokolde ise marketten alınan obje haritaya yerleştirilmek isteniyordur. Bu durumda o objeye ait bina tipide gönderilir. Bu bina tipi ile veri tabanına giriş yapılır. Sonuç olarak "I" protokolü gönderilir.
- I protokolü: Server ilgili objeye ait tüm bilgileri Flash'a göndermek için kullanır. Bu protokol "I|Bina id|X|Y|Action ya da Bitiş zamanı|Bina Tipi" şeklindedir. Binanın id sini objenin koordinatlarını ve bina tipini içerisinde bulundurur. Ayrıca eğer ev ilk defa kurulmuşsa Action kısmında -1 değeri gönderilir. İlk defa kurulmamışsa bir sonraki kiranın bitiş zamanı gönderilir. Eğer

bu bir tarla ise burada tarlanın hasat zamanı ve çürüme zamanı gibi bilgiler gönderilir.

- Aksiyon Protokolü: "A|Bina id|Aksiyon No" şeklindeki protokoldür. Bu protokol Flash tarafından Server'a gönderilir. Öncelikle bina id ile veri tabanından o objeye ait bilgiler çekilir. Bina tipine bakılarak tarla mı olduğu yoksa bina mı olduğu bilgisine göre veriler değerlendirilir. Veri tabanından çekilen bilgi doğrultusunda Flash'a "I" protokolü gönderilir. Ve veri tabanında gerekli değişiklikler yapılır.

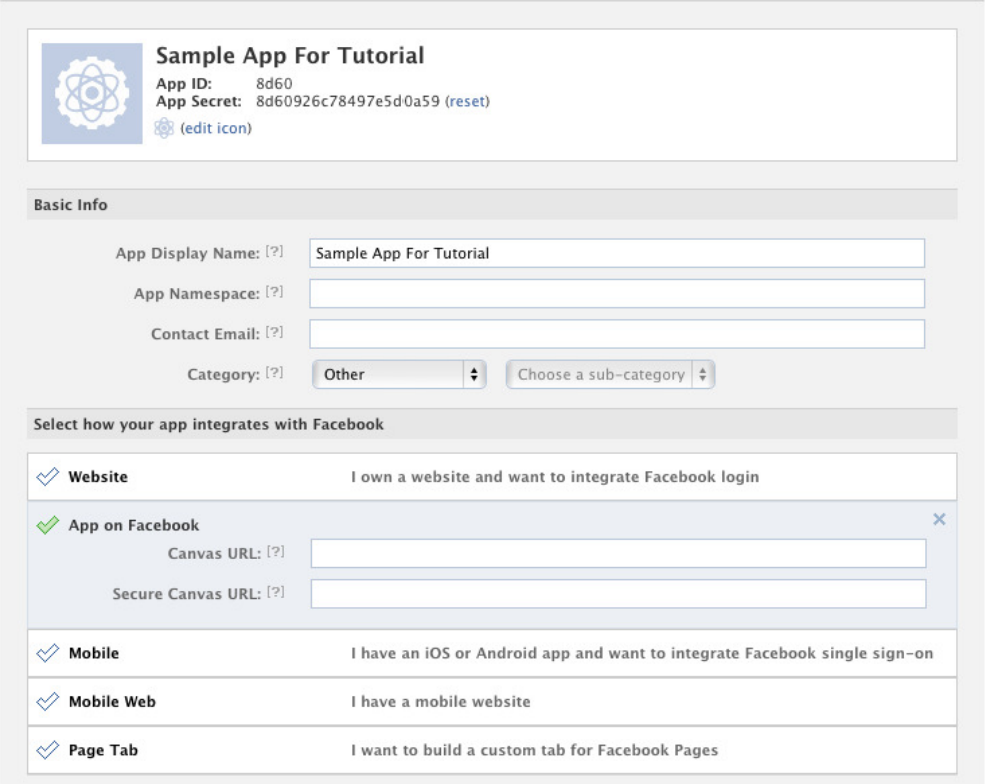
9. Facebook Uygulaması

Php Facebook uygulamasının kurulumunda kullanılmaktadır.

9.1. Uygulama Ayarları

Uygulamanın facebook'a bağlanabilmesi için gereken ilk işlem facebook hesabına developer app kurulumudur.

Şekil 9.1: Developer application kurulumu



Şekil 9.2: Uygulama ayarları

Developer application kurulumundan sonra yeni bir uygulama oluştur butonuyla yeni bir uygulama oluşturulur şekil 9.2'deki ekranda görülen uygulamanın ismi hangi url üzerinde çalışacağı gibi ayarlar yapılır. Secure canvas url için ssl sertifikalı bir adres kullanılmalıdır.

Şekil 9.3: Boyutlandırma ayarları

Şekil 9.3 te görülen ayarlar facebook içerisinde görünecek uygulamamızın genişlik ve yüksekliğinin dinamik olup olmayacağıdır.

9.2. Php Kütüphanesi

9.1'deki işlemlerden sonra facebook'un uygulamamız application id, canvas linki ve secret anahtarı vermektedir. Bu bilgiler facebook php kütüphanesi indirilerek değişkenlere atanır ve bu sayede facebook graph api ile bağlantı kurulması sağlanır. Yeni bir Facebook nesnesi tanımlayarak uygulama ID'sını ve secret key'i burada faceboook nesnesi içerisine parametre olarak verilmektedir. Bu nesne örneğini kullanarak oyuncunun facebook ID'si gibi bilgileri elde edilebilmektedir. Bunun yanı sıra uygulamaya ilk defa mı girecek yoksa daha önceden bu uygulamayı açmış mı diye kontrol ederek oyuncudan duruma göre gerekli izinlerin alınacağı bir pencere çıkartılmaktadır. Bu pencerede isteğe göre kişisel bilgiler, paylaşım izni, mail adresine erişim gibi kişisel veriler kullanıcıdan istenebilir. Son olarak c#'ta da anlatılan hash işlemi burada da uygulanmaktadır. Uygulama içerisinde iframe oluşturularak hash ve facebook ID'si servera gönderilerek kullanıcıya ait oyun hesabına giriş yapılması ve oyun ekranının gösterilmesi sağlanır. (Zuckerberg, 2012)

Şekil 9.4: Davet sayfası

Uygulamanın bir diğer önemli parçası davet sayfasıdır. Davet sayfası uygulamanın daha çok kişiye ulaşması bakımından önemlidir. Bu sayfada Facebook Query Language kullanılarak kullanıcının uygulamamıza üye olmamış olan arkadaşları sorgulanır. Sorgudan dönen kullanıcının arkadaşlarına ait facebook kullanıcı numaraları ve isim bilgileri htmlvars ile flasha aktarılır. Kullanıcı davet etmek istediği arkadaşlarını seçip davet tuşuna bastığında sayfa içerisindeki facebook javascript kütüphanesi flash tarafından tetiklenir. Bu aşamadan sonra facebook kullanıcının arkadaşlarını davet isteği gönderir. (Ben Kirman, 2009)

10. SONUÇ

Flash üzerinde çalışan bir takım 3 boyutlu motorlar bulunmaktadır. Bunlardan bazıları "Alternativa3D", "Away3D", "Papervision" ve "Sandy3D"dir. Ancak bunların içerisinde en fazla poligon sayısına ulaşabilen "Alternativa3D"dir. Bu nedenle diğer oyun motorları yerine "Alternativa3D"nin seçmesi performans ve kalite bakımından daha başarılı sonuçlar vermektedir.

Flash içerisinde kullanılan değişkenler kolaylıkla değiştirilebildiği için oyunun sadece Flash üzerinde çalışması güvensiz bir ortam oluşturmaktadır. Bu nedenle kullanıcının her hareketini takip eden ve oluşan bu aksiyonlara cevap veren bir server programlanmıştır. Bu sayede oyuncuların Flash'a müdahale ederek para miktarlarını istedikleri gibi değiştirmeleri engellemiştir.

Alternativa3D özellikle üç boyutlu objelerin oyuna aktarılması sırasında 3DS Max ya da Blender gibi programlara önem vermektedir. Bu durumu web sitesindeki 3DS Max için yazılmış "plugin"lerden ve "tutorial"lerde özellikle Blender programının kullanılmasından anlayabiliriz. Ancak çalıştığımız platform "Windows" olduğundan uygulamamızda 3DS Max tercih edilmiştir.

.net ile php haberleşmesi sırasında oyuncuların farklı bir oyuncunun hesabına girmesini önlemek amacıyla "md5 hashing" algoritması kullanılmıştır.

Gpu desteği oyunun 3 boyutlu olmasına rağmen iyi bir performans göstermesini sağlamıştır. Bunun yanında 3 boyutlu modellerin ve bu modellere ait kaplamaların dosya boyutlarının fazla olması yükleme sürelerinin uzamasına yol açmaktadır. Bu durum şuan için dezavantaj olarak görülebilir ancak ülkemizdeki internet hızının gelişimi göz önüne alındığında bu dezavantajda yakın gelecekte ortadan kalkarak hem sosyal hemde gelişmiş grafiklere sahip online oyunlar hazırlanmasına olanak sağlayacaktır.

Facebook ortamında çok yoğun bir kullanıcı kitlesi olduğu düşünüldüğünde yapılan uygulamanın böylesine geniş bir ortamda daha hızlı yayılacağı aşikârdır.

11. KAYNAKLAR

Independent, Accurate Application Metrics and Trends from Inside Network. (2012). 4 20, 2012 tarihinde http://www.appdata.com/leaderboard/. adresinden alındı

Ben Kirman, S. L. (2009). Gaming on and off the Social Graph: The Social Structure of Facebook Games. *International Conference on Computational Science and Engineering*. Lincoln, UK: Lincoln Social Computing Research Centre University of Lincoln.

David Huffaker, J. (. (2009). The Social Behaviors of Experts in Massive Multiplayer Online Role-playing Games. *International Conference on Computational Science and Engineering*. USA.

Edward F. Gehringer, M. D. (2009). Work in Progress - Game Mechanics and Social Networking for Co-Production of Course Materials. *39th ASEE/IEEE Frontiers in Education Conference*. San Antonio, TX: North Carolina State University.

Games, A. I. (2011). *An Insider's Guide to Designing and Developing the World's Greatest Video Games.*

GÖZÜDELİ, Y. (2011). *Yazılımcılar İçin SQL Server 2008 & Veritabanı Programlama.* İstanbul: Seçkin Yayıncılık.

Henderson, K. (2003). *SQL Server Depolanmış Yordamları, XML VE HTML.* İstanbul: Alfa Yayınları.

Hou, H.-T. (2010). Applying Lag Sequential Calculation and Social Network Analysis to Detect Learners' Behavioral Patterns and Generate Automatic Learning Feedback-Scenarios for Educational MMORPG Games. *IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*. Taipei, Taiwan.

John Sharp, J. J. (2005). *ADIM ADIM MICROSOFT VISUAL C# .NET.* Ankara: Arkadaş Yayınevi.

Kirman, B. (2010). Improving social game engagement on Facebook through enhanced socio-contextual information. *Proceedings of the ACM Conference on Human Factors in Computing Systems.* Atlanta.

Kyoung-Jin Park, A. Y. (2010). Social Network Approach to Analysis of Soccer Game. *International Conference on Pattern Recognition.* Columbus, Ohio: The Ohio State University.

Reddy, Y. B. (2009). Role of Game models in Social Networks. *International Conference on Computational Science and Engineering* (s. 1132). Grambling, USA: Grambling State University.

Wohn, D. Y., Lampe, C., Wash, R., Ellison, N., & Vitak, J. (2011). The “S” in Social Network Games: Initiating, Maintaining, and Enhancing.

Zuckerberg, M. (2012, 5 20). *https://developers.facebook.com/*. 04 15, 2012 tarihinde Facebook: https://developers.facebook.com/ adresinden alındı

12. ÖZGEÇMİŞ

1985 yılında İstanbul'da doğdu. İlk, orta ve lise öğrenimini İstanbul'da tamamladı. Nişantaşı Nuri Akın Lisesi'ni bitirdikten sonra Dumlupınar Üniversitesi Makina Mühendisliği bölümünden 2007 yılında mezun oldu.