# Oyun Programlama
# (Vize)

* Oyun; gelişimi bilgi teknolojilerinin, ekran kartlarının gelişimine bağlıdır. İşlemler hızlı ve gerçek zamanlı olmalıdır.

Ekran kartı işlemcisi
Ekran kartı hafızası
Ekran kartı dönüştürücüleri
} önemli!

* Ekran kartı programlanabilirse (CUDA tabanlı, string tabanlı) oyun programlama için daha iyidir.

* Heterojen programlama; günümüzde kullanılan bir tekniktir. Hem ekran kartının hem de CPU'nun bileşenleri kullanılır. (OpenCL)

* Oyun motorları; bilgisayar yazılımdan uzaklaştırarak daha üst seviyeye taşır. (Game Engine) Çoğunlukla C, C++ tabanlı.

* Directx; oyun kartındaki işlemleri ekran kartına yükler.

* Oyun Platformları

- PlayStation 2 (Cpu: 300 MHz)
- Microsoft Xbox (Cpu: 733 MHz, 8gb hard drive)
- Nintendo Game Cube (405 mHz)
- Nintendo Game Boy Advance
- PC

* Oyun Yazım Döngüsü *

1) Konsept
2) Senaryo
3) Karakter Modelleme
4) Karakterler arası etkileşim (durum diyagramları)
5) Üretilmesi
6) Test işlemi

: Oyun Programlama; bilgisayarlı grafiğin alt dalıdır Grafik, matematik e fizik bilgisi gerektirir.

) Animasyon işlemlerinin daha üst düzey programlamasıdır.

) Kullanıcı ile etkileşim söz konusudur. Özellikle gerçek zamanlı işlemlerde ilmeli programlama tekniklerini ne yapmak zordur.

) Hızlı bir şekilde işlem yapmayı sağlayan bilgisayar donanımına müdahele edebilen yazılımlar kullanılır. Bunlara kısaca oyun motoru ılır. Çoğunlukla C, C++ tabanlı olmakla birlikte Java ve python estekliidir. Kendi içinde script dilini kullanır

)Gerçek dünyadaki fizik işlemleri ile ilgili hareketleri, davranışları odellemek için (matematiksel ve fiziksel model) fizik motorları da ullanılır.

) Oyun geliştirilirken aşağıdaki adımlar takip edilir.

- Senaryo oluşturma
- Modelleme (ortam, karakter, ..)
- Karakterler arası etkileşim için durum diyagramı
- Oyun içindeki etkileşimler için basit veya yüksek seviyede yapay zeka teknikleri

) İyi bir oyun thread üzerinde çalışır. İşlemleri hızlandırmak için kran kartının işlemcisine müdahele edebilmektedir.

) Oyun tasarımı ile oyun geliştirme farklı konulardır. Oyun tasarımı erçek dünyadaki çeşitli davranışların (ekonomi, siyasal, askeri, ..) bilgisayar rtamına aktarılmasıdır.

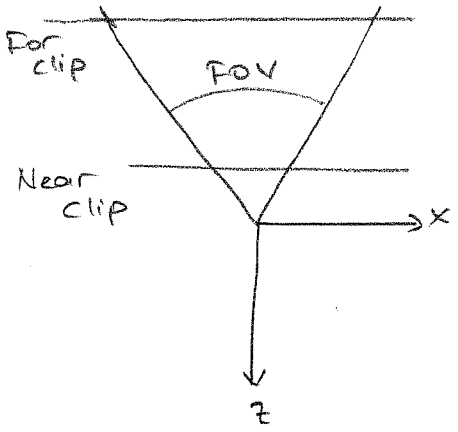
Sahne Yönetimi (Scene Management)

' Ortamın modellenmesidir. Ortamdaki nesnelerin ve karakterin hangi sıra, ı ve hangi detay ile çalışılacağını ortaya koymasıdır.

Temel işlem "renderer" işlemidir. Yani derinlik kazandırmadır. Karakterin eya kameranın pozisyonuna göre bu işlemin dinamik yapılması gerekir. ndering işleminde ağırlıklı olarak üçgensel, dörtgensel çokgenler kullanılır. ahne yönetimi önemli bileşenleri karakterler içinde lazım olanı ekip çıkarmak, LOD detayı seviyesinde çalışmak, çizim sırasına dikkat tmek ve bu işlemler için ekranı (görünebilir ekranı) sayfalara eklinde kullanmaktır.

## Culling

a) Karakterler içinde gereten karakteri, nesneyi çizme işlemi culling olarak adlandırılır. Bunu tanım etmek için kameraların ve nesnelerin pozisyonu temel kriterlerdir.

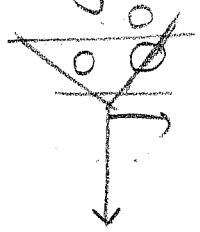



* Near Clip: kameranın görmeye başladığı uzaklık değeri
* For Clip: kameranın algılayabileceği uzaklık değeri
* FOV: bakış alanı açısı, kamera merceğinin hangi açı ile görebileceği değer

## Bounding Volumes

Anlık çizim işleminde nesneler ile ilgili 3 durum söz konusudur. Bu durumu kameranın bakış açısı ortaya koyar.

- görülebilen nesneler
- görülemeyen nesneler
- yarı görünen nesneler



* yarı görülebilen nesneler için gerek 2 boyutta gerek 3 boyutta kırpma algoritmaları gerektirir. 2 boyutta çalışıldığı zaman çember, dikdörtgen gibi yapılar kullanılır. 3 boyutta ise küre, silindir, prizma gibi yapılar kullanılır.

## Optimal Enclosing Sphere

Nesnelerin işgal ettiği alanı bulmak için optimum yöntem kullanmak faydalıdır. Bunun amacı nesnenin/nesnelerin tam olarak işgal ettiği alanı bulmaktır. Bu işlemden sonra test işlemine tabi tutulurlar. Test işleminde hangi nesnelerin görülebilir, hangilerin görünmez olduğunu çalışılan tüm düzlemlerde tespit edilir. Test işlemi demek birden fazla bakış açısına göre kırpma algoritmasının çalıştırılmasıdır.

## Görülebilir Alan / Nesneler

) Bir nesnenin görülebilir olması, kamera pozisyonu ile ilgilidir. Kameranın
ı olması ve açısının pozisyonunun doğru tanımlanmasını gerektirir.
) Nesne tarafında ise görülebilir olması o nesnenin detay modellenmesi
e ilgilidir. Kameraya yakın nesneler üzerinde detay çalışılır. Uzak
esneler üzerinde kaba çalışılır.
) Ön taraftaki nesneler üzerinde donanım bağımlı olarak, oyun tipine
e senaryosuna bağlı olarak gerçekleştirilecek detay çalışması optimum
gerektir. Çoğu zaman sayısal değer vermek mümkün değildir.

## Level of Detail (Detay Seviyesi)

ön taraftaki nesnelerin eğrisellikleri, kavisleri, türev geçtiklerinin hızlı
ı da yoğun olduğu noktaların ortaya konulması LOD işlemidir. Bu işlem
ın önce nesnenin kaba modeli çıkartılır. Daha sonra istenen hassasiyete
öre detaylar ortaya konulur. Bunun için en yaygın kullanılan üç
öntem vardır

) Progressive Meshes (Yaygın Yöntem): Minimum sayıda nokta tutulur.
kameraya yaklaştıkça detaylar artar.
) Alt bölmeleme (Subdivision): Nesnenin özellikle doğal ağaç, bulut gibi
apıların modellenmesinde kullanılır. Ortanın kaba noktaları üretilir.
oğunlukla üçgensel yapı ile çalışılır. Temel çalışma noktası, bir
çgeni dört üçgen yapmaktır.

OT: İşlemler hem CPU hem de GPU (ekran kartı işlemcisi) üzerinde
erçekleşmeli. Thread şeklinde çalışma yapılmalıdır, threadler mümkün olduğu
adar çok küçük program parçacıklarını GPU'da çalıştırmalıdır.

Sahne yönetimi çizim sırası

Modelleme işlemi
Tüm modellerden nesne/nesneleri alkama işlemi
LOD işlemi
Bu modeller üzerinde parlaklık işlemi
" " " çeşitli etkilerin sağlanması
Modellerin, kameranın pozisyonuna göre sıralanması (Z-Buffer Alg.)
Gölgeli kameraların yerleştirilmesi

## Çarpışma Testi

*Grafik veya oyun programlama tekniklerinin matematiksel açıdan yük gerektiren konularından biridir. İki boyutlu veya üç boyutlu uzayda nesne kesişimlerinin, nesne karakterine göre kesişip kesişmeyeceğinin tahmin edilmesi yöntemidir.

*İki boyutlu uzayda çözüm çoğu zaman kolaydır. Nesne sayısı arttıkça o zaman sıkıntılar ve çözülmesi gereken denklem sayısı artar.

*Üç boyutlu uzayda çözüm yapılırken iki boyutlu uzaya dönüşüm yapmak bir yöntemdir.

*Veri yapısı olarak özellikle ağaç yapıları kullanılır. İşlemleri hızlandırmak için çözülen denklem takımları asıl ifadeden çıkarılır.

## Fizik Simulasyonu

Gerçek dünyadaki fizik kurallarının özellikle hareket ve yer değiştirmeye bağlı olan denklemlerin bilgisayar ortamında modellenmesi ve kullanılması işlemlerini belirtmek için fiziksel simülasyon yapılır. Oyun motorlarında kendisinin kullandığı fizik motorları ve genel anlamda geliştirilmiş olan fizik motorları bu işlemleri açık kaynak kodu ya da ticari amaçlı olarak sağlar.

* Özellikle karakterlerin hareketinin modellenmesinde inverse kinematic denklemler kullanılır. Robot kolunun uzanabileceği ve genel anlamda bu kolun hareketinin modellenmesinde belli bir dif. denkleme bağlı olarak çalışan inverse kinematic'lerin çözülmesi gerekir. Hesaplamada yapılan işlem vektör ve açı hesabıdır. Bu işlem karakterin eklem yapılarına bağlı olarak ne tür bir alan kullanacağının hesabında da geçerlidir.

## Character Animation

Robot kolu ve eklemlerden oluşan yapıların animasyonunda eklemlerin serbestlik derecesi D.O.F önemli etkendir. Tek boyutlu ya da tek dereceli serbestliğe sahip yapılar tek bir eksende bir yönde hareket kabiliyetine sahiptir. Maximum 6 serbestlik derecesi mümkündür.

→

serbestlik dereceli bir yapı bütün eksenlerde hareket yeteneğine sahiptir. Özellikle robot kolu gibi animasyonlarda bu serbestlik derecesi hesaplamalarının düzgün yapılması gerekir.
Bir karakterin robot kolunun hareketini etkileyen önemli diğer faktör de eklem yapısıdır.

## Free Form Deformation

Düzey modellemenin ve deformasyon hesabının en güncel konusudur. Üçgenli parçalar üzerinde hesaplama işlemine dayanır. Nesneyi modellerken kullanılan üçgenin şekli (dikdörtgen-üçgen) önemlidir. Ne kadarlık deformasyon olacağının bulunması bu parçalar üzerinde bilinen bir diferansiyel denklemin çözümünü gerektirir. Çoğu zaman deneme-yanılma yolu tercih edilir. Mühendislik analiz tarafında ise kesinlikle diff. denklem çözülmelidir.

## OSI Katmanları

1-) Uygulama: Kullanıcı programlarının ağ kaynaklarına erişimini sağlar.

2-) Sunum: Verileri uygulama katmanına sunarken veri üzerinde kodlama ve biçimlendirme yapar.

3-) Oturum: Aynı bilgisayar üzerinde farklı oturumların açılması hizmeti verirken güvenliğini sağlar.

4-) Taşıma: Bu katman 5-7 ve 1-3 katmanları arasında köprü görevi yapar. Üst katmandan aldığı veriyi parçalar alta verir. Alttan aldığı veriyi birleştirip üste verir.

5-) Ağ: Veri paketlerinin yönlendirme işlemi yapar.

6-) Veri: Ağ katmanından aldığı verilere hata kontrol bitleri ekleyerek çerçeve haline getirir.

7-) Fiziksel: Verilerin fiziksel olarak gönderilmesinden ve alınmasından sorumlu.