



cub3D

miniLibX ile ilk RayCaster'ım

*Özet: Bu proje, ilk FPS oyunu olan dünyaca ünlü oyun Wolfenstein 3D'den esinlenerek oluşturulmuştur. Ray-casting(Işın Takibi) konusunu keşfetmenizi sağlayacaktır.*

*Amacınız, bir labirentin içinde yolunuzu bulmak için dinamik bir bakış açısı oluşturmanız.*



*Versiyon: 9*

# İçindekiler

I	Önsöz	2
II	Hedefler	3
III	Genel Talimatlar	4
IV	Zorunlu kısım - cub3D	6
V	Bonus kısım	10
VI	Örnekler	11



# Bölüm I

## Önsöz

Id Software'de John Carmack ve John Romero tarafından geliştirildi, 1992 yılında Apogee Software tarafından yayınlandı. Wolfenstein 3D, oyun tarihindeki ilk gerçek "First Person Shooter" oyunu.



Figure I.1: John Romero (sol) and John Carmack (sağ) poster için poz verirken.

Wolfenstein 3D, Doom (Id Software, 1993), Doom II (Id Software, 1994), Duke Nukem 3D (3D Realm, 1996) ve Quake (Id Software, 1996) ve daha nicelerinin atasıdır.

Şimdi tarihi yeniden yazma sırası sende...

# Bölüm II

## Hedefler

Bu projede kullanacağınız diğer ilk yıl projelerine benzer:  
C ve temel algoritmaların kullanımı, bilgi araştırması vb.

Bir grafik tasarım projesi olarak, cub3D pencereler, renkler, boşlukları doldurma vb.  
alanında gelişmenizi sağlayacaktır.

Sonuç olarak cub3D, ayrıntıları anlamak zorunda kalmadan matematiğin eğlenceli ve  
pratik uygulamalarını keşfetmek için olağanüstü bir oyun alanıdır.

İnternette bulunan sayısız belgenin yardımıyla, matematiği zarif ve verimli algorit-  
malar oluşturmak için bir araç olarak kullanacaksınız.



Projeye başlamadan önce orijinal oyunu denemenizi öneriyoruz:

<http://users.atw.hu/wolf3d/>

# Bölüm III

## Genel Talimatlar

- Projeleriniz C programlama dilinde yazılmalıdır.
- Projeleriniz Norm'a uygun olarak yazılmalıdır. Bonus dosyalarınız/fonksiyonlarınız varsa, bunlar norm kontrolüne dahil edilir ve bu dosyalarda norm hatası varsa 0 alırsınız.
- Tanımlanmamış davranışlar dışında sizin fonksiyonlarınız beklenmedik bir şekilde sonlanmamalıdır (Segmentasyon hatası, bus hatası, double free hatası, vb.) . Eğer bunlar yaşanırsa s 0 alırsınız.
- Heap'de ayırmış olduğunuz hafıza adresleri gerekli olduğu durumlarda serbest bırakılmalıdır. Hiçbir istisna tolere edilmeyecektir.
- Eğer verilen görev **Makefile** dosyasının yüklenmesini istiyorsa, sizin kaynak dosyalarınızı **-Wall**, **-Wextra**, **-Werror**, flaglarını kullanarak derleyip çıktı dosyalarını üretecek olan Makefile dosyasını oluşturmanız gerekmektedir. Makefile dosyasını oluştururken cc kullanın ve Makefile dosyanız yeniden ilişkilendirme yapmamalıdır (re-link).
- **Makefile** dosyanız en azından **\$(NAME)**, **all**, **clean**, **fclean** ve **re** kurallarını içermelidir.
- Projenize bonusu dahil etmek için Makefile dosyaniza **bonus** kuralını dahil etmeniz gerekmektedir. Bonus kuralının dahil edilmesi bu projenin ana kısmında kullanılması yasak olan bazı header dosyaları, kütüphaneler ve fonksiyonların eklenmesini sağlayacaktır. Eğer projede farklı bir tanımlama yapılmamışsa, bonus projeleri **\_bonus.{c/h}** dosyaları içerisinde olmalıdır. Ana proje ve bonus proje değerlendirmeleri ayrı ayrı gerçekleştirilmektedir.
- Eğer projeniz kendi yazmış olduğunuz **libft** kütüphanesini kullanmanıza izin veriyorsa, bu kütüphane ve ilişkili **Makefile** dosyasını proje dizinindeki **libft** klasörüne ilişkili **Makefile** dosyası ile kopyalamanız gerekmektedir. Projenizin **Makefile** dosyası öncelikle **libft** kütüphanesini kütüphanenin **Makefile** dosyasını kullanarak derlemeli ardından projeyi derlemelidir.
- Test programları sisteme yüklenmek zorunda değildir ve puanlandırılmayacaktır. Buna rağmen test programları yazmanızı şiddetle önermekteyiz. Test programları

sayesinde kendinizin ve arkadaşlarınız projelerinin çıktılarını kolaylıkla gözlemlayabilirsiniz. Bu test dosyalarından özellikle savunma sürecinde çok faydalananızsınız. Savunma sürecinde kendi projeleriniz ve arkadaşlarınızın projeleri için test programlarını kullanmakta özgürsünüz.

- Çalışmalarınız atanmış olan git repolarına yüklemeniz gerekmektedir. Sadece git reposu içerisindeki çalışmalar notlandırılacaktır. Eğer Deepthought sizin çalışmanızı değerlendirmek için atanmışsa, bu değerlendirmeyi arkadaşlarınızın sizin projenizi değerlendirmesinden sonra gerçekleştirecektir. Eğer Deepthought değerlendirme sürecinde herhangi bir hata ile karşılaşılırsa değerlendirme durdurulacaktır.

## Bölüm IV

### Zorunlu kısım - cub3D

Program adı	cub3D
Teslim edilecek dosyalar	Tüm dosyalarınız
Makefile	all, clean, fclean, re, bonus
Argümanlar	*.cub formatında bir harita
Harici fonksiyonlar.	<ul style="list-style-type: none"><li>• open, close, read, write, printf, malloc, free, perror, strerror, exit</li><li>• Math kütüphanesinin tüm fonksiyonları (-lm man man 3 math)</li><li>• MinilibX'in tüm fonksiyonları</li></ul>
Libft kullanılabilir mi?	Evet
Açıklama	Birinci şahıs bakış açısıyla bir labirentin içinin "gerçekçi" bir 3D grafik gösterimini oluşturmalısınız. Bu gösterimi daha önce bahsedilen İşın Takibi(Ray Casting) ilkesini kullanarak oluşturmalısınız.

Kısıtlamalar aşağıdaki gibidir:

- **miniLibX** kullanmanız gerekiyor. Ya işletim sisteminde bulunan sürümü ya da kendi kaynaklarındaki sürümü kullanın. Kaynaklarla çalışmayı seçerseniz, libft için yukarıda Ortak Talimatlar bölümünde yazılanlarla aynı kuralları uygulamanız gereklidir.
- Pencere kontrolünüz başka bir pencereye geçme, küçültme vb. durumlarda akıcı olmalıdır.

- Duvarın hangi tarafa baktığını (Kuzey, Güney, Doğu, Batı) bağlı olarak değişen farklı duvar dokuları sergileyin (seçim sizin).
- Programınız zemin ve tavan renklerini iki farklı renge ayarlayabilmelidir.
- Program görüntüsü bir pencerede gösterir ve aşağıdaki kurallara uyar:
  - Klavyenin sol ve sağ ok tuşları labirentte sola ve sağa bakmanızı izin vermelidir.
  - W, A, S ve D tuşları, bakış açısını labirentte hareket ettirmenize izin vermelidir.
  - ESC'ye basmak pencereyi kapatmalı ve programdan temiz bir şekilde çıkmalıdır.
  - Pencere çerçevesindeki kırmızı çarpı işaretine tıklamak pencereyi kapatmalı ve programdan temiz bir şekilde çıkmalıdır.
  - **minilibX** kullanılması şiddetle önerilmektedir.
- Programınız ilk argüman olarak bir harita dosyasını .cub uzantısıyla almalıdır.
  - Harita 6 olası karakterden oluşmalıdır: Boş alanlar için **0**, Duvarlar için **1**, ve kullanıcının başlangıç pozisyonu ve yeniden doğma yönü için **N,S,E** veya **W** Basit ve geçerli bir harita örneği:

```
111111
100101
101001
1100N1
111111
```
  - Harita kapalı veya duvarlarla çevrili olmalı, değilse program bir hata vermelidir.
  - Harita içeriği dışında, her öğe türü bir veya daha fazla boş satırla ayrılabilir.
  - Her zaman en son olması gereken harita içeriği dışında, her öğe türü dosyada herhangi bir sırada ayarlanabilir.
  - Harita dışında, bir ögeden gelen her bilgi türü bir veya daha fazla boşlukla ayrılabilir.
  - Harita, dosyada göründüğü gibi ayırtılmalıdır. Boşluklar haritanın geçerli bir parçasıdır ve işlemek size kalmıştır. Harita kurallarına uyduğu sürece her tür haritayı ayırtılabilmeniz gereklidir.

- o Her ögenin (harita hariç) ilk bilgileri, tür tanımlayıcısıdır (bir veya iki karakterden oluşur), ardından her nesne için kesin bir sırayla tüm özel bilgiler gelir, örneğin:

- \* North texture(Kuzey dokusu):

```
NO ./path_to_the_north_texture
```

- Tanımlayıcı: **NO**
- North texture dosya yolu

- \* South texture(Güney dokusu):

```
SO ./path_to_the_south_texture
```

- Tanımlayıcı: **SO**
- South texture dosya yolu

- \* West texture(Batı dokusu):

```
WE ./path_to_the_west_texture
```

- Tanımlayıcı: **WE**
- West texture dosya yolu

- \* East texture(Doğu dokusu):

```
EA ./path_to_the_east_texture
```

- Tanımlayıcı: **EA**
- East texture dosya yolu

- \* Zemin rengi:

```
F 220,100,0
```

- Tanımlayıcı: **F**
- R,G,B renkleri, [0,255] aralığında: **0, 255, 255**

- \* İç kaplama rengi:

```
C 225,30,0
```

- Tanımlayıcı: **C**
  - R,G,B renkleri, [0,255] aralığında: **0, 255, 255**

- Zorunlu kısmın minimalist bir örneği:

```
NO ./path_to_the_north_texture
SO ./path_to_the_south_texture
WE ./path_to_the_west_texture
EA ./path_to_the_east_texture

F 220,100,0
C 225,30,0

    1111111111111111111111111
    1000000000110000000000001
    1011000001110000000000001
    1001000000000000000000001
1111111101100001110000000000001
100000000011000011101111111111
111101111111101110000010001
11110111111110111010010001
11000000110101011100000010001
10000000000000001100000010001
1000000000000000110100010001
1100000111010101111011110N0111
11110111 1110101 101111010001
11111111 1111111 111111111111
```

- Dosyada herhangi bir yanlış yapılandırma ile karşılaşılırsa, program düzgün bir şekilde kapanmalı ve seçtiğiniz açık bir hata mesajının ardından "Error\n" doldurmalıdır.

# Bölüm V

## Bonus kısım



Bonuslar, yalnızca zorunlu bölümünüz MÜKEMMEL ise değerlendirilecektir. MÜKEMMEL ile doğal olarak şunu kastediyoruz: eksiksiz olması, yanlış kullanımlar vb. gibi kötü hatalar olması durumunda bile düzgün çalışması. Bunun anlamı, zorunlu bölümünüz, derecelendirme sırasında TÜM puanları almazsa, bonuslarınız tamamen YOK SAYILACAKTIR.

Bonus listesi:

- Duvar çarpmaları
- Mini harita sistemi.
- Açılıp kapanabilen kapılar.
- Animasyonlu spritelar.
- Mouse yardımıyla bakış açısını değiştirme.



İlerde daha iyi oyunlar geliştirebileceksiniz. 0 yüzden çok fazla zaman harcamayın!



Değerlendirmeniz sırasında kullanıcıları serbest olduğu sürece, bonus bölümünü tamamlamak için başka fonksiyonları kullanmanıza veya haritaya semboller eklemenize izin verilmektedir. Ayrıca, istenen harita dosyası formatını ihtiyaçlarınıza göre değiştirebilirsiniz. Akıllı davranışın!

# Bölüm VI

## Örnekler



Figure VI.1: RayCasting kullanılan orijinal *Wolfenstein3D* oyunu.

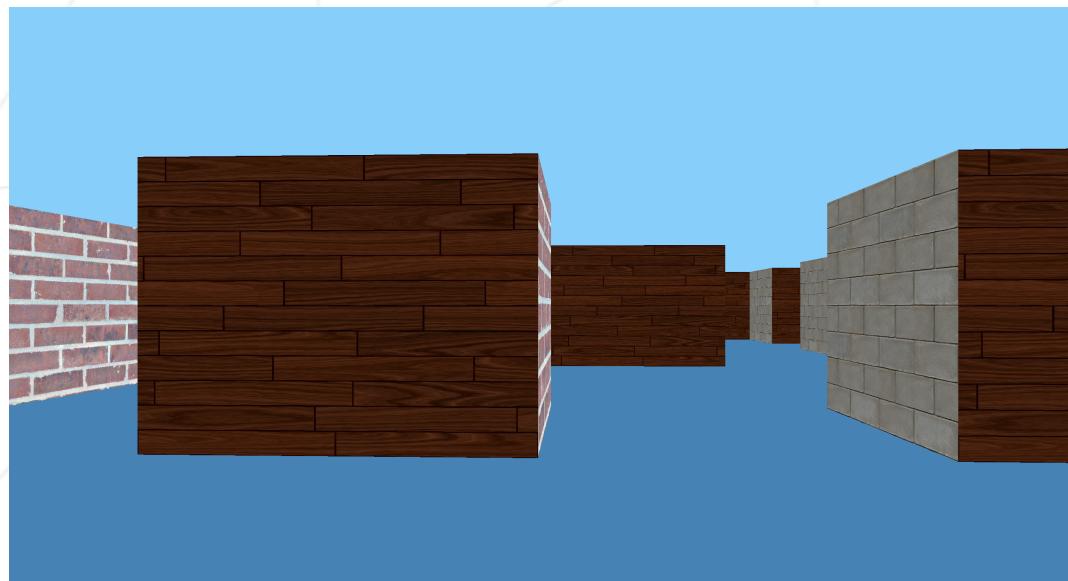


Figure VI.2: Zorunlu bölüme göre projenizin nasıl görünebileceğine dair örnek.



Figure VI.3: Mini harita, zemin ve tavan dokularıyla beraber hareketli bir Sonic sprite'ı içeren bonus parçası örneği.



Figure VI.4: HUD, sağlık cubuğu, gölge efekti ve ateş edebilen silah içeren bir başka bonus örneği



Figure VI.5: Seçtiğiniz bir silah ve tavana bakan oyuncu ile bir başka bonus oyunu örneği