SDL ile Oyun Programcılığı

Sinan Ateş

Nisan, 2016

İçindekiler

1 Giriş

Bilgisayar kullanmaya hemen hemen hepimiz ilk olarak oyun oynayarak başlamışızdır. Biraz daha ileri gittiğimizde artık bir programlama dili öğreniriz ve konsol ekranında çalışan programlar yazarız. Bu programlardan sonra bir seviye daha atlayarak Grafik Arayüzü olan programlar yazarız. Bütün bu süreç içerisinde bazılarımızın içerisinde hep hayallerindeki bir oyunu yazmak vardır. Bu yazı dizisinde kendi oyunlarımızı yazabilmek için gerekli bir alt yapıyı oluşturmaya çalışacağız.Umarım bu yazı dizisi ilginizi çeker.

Burada yazacağımız dersler 2 Boyutlu basit oyunlar yapmamız için bir temel oluşturacaktır. Bir seviye atlayarak 3 Boyutlu oyunlarda yazabilirsiniz. Tahmin edebileceğiniz gibi yazacağınız oyunlar muhteşem grafiklere sahip, hatalardan arındırılmış, gerçekçi oyunlar olmayacaktır. Şunu söyleyebilirim ki yazdığınız oyunlar sizi fazlasıyla mutlu edecektir. Aslında oyun yazmak bir ekip işidir. Bu ekipte programcılar dışında senaryo, grafik, ses gibi bir çok alandan insan çalışır. Ayrıca programcıları bile kendi aralarında ayırabiliriz. Oyun içi mekanikleri yazanlar, grafik motoru, fizik motoru, AI(yapay zeka)…. programcıları gibi farklı sınıflara ayırabiliriz. Grafik motoru ile ekranda grafikleri gösterebilmek, fizik motoru ile oyun içerisinde oluşturduğumuz ortamın fizik kuralları ile uyumlu olabilmesini sağlamak için kullanılır. Oyunlarda düşmanımızında biraz mantıklı hareket etmesini isteriz değil mi? Bunu da yapay zeka ile sağlarız. Bunları bu şekilde ayrı ayrı yazmanın avantajı, bu kodların bir başka oyunda veya oyunumuzun devam serilerinde tekrar kullanılabilmesini sağlar. Eğer bütün herşey baştan yazılmaya kalkışılsa kaybedilecek zamanı tahmin edebiliyor musunuz? Ayrıca bu yazdığımız kodlarla bir başkasıda kendi oyununu yapabilir. Bu yüzden bunlar oyun motorları diye adlandırılır. Piyasa da bir çok açık kaynak ve ücretsiz oyun motorları mevcut. Belki bu işe meraklıysanız bu motorlarla içli dışlı olabilir, oyunlarınızı biraz daha güzelleştirebilirsiniz.

Oyun yapımı hakkında bu kadar önbilgiden sonra şimdide kullanacağımız kütüphane hakkında biraz bilgi verelim. Programlama dili olarak C++ kullanacağız çünkü C++ bu endüstride kabul görmüş bir dil. Hız olarak en verimli dil olduğu söyleniyor. Oyun yapımı için C++ ile birlikte kısaca SDL diye bilinen bir kütüphane kullanacağız. SDL aslında bir kısalmadır ki açılımı şu şekildedir. Simple DirectMedia Layer. Aşağıdaki yazıyı aynen Wikipediadan buraya aktardım.

> SDL (Simple DirectMedia Layer), ilk olarak 1998 yılında Sam Lantinga tarafından C programlama dili ile yazılımş, çapraz platform, özgür ve açık kaynak kodlu yazılım çoklu ortam kütüphanesi. Birçok platformda değişikliğe gerek duymadan grafik, ses, klavye, fare etkileşimi sunan bir arabirim niteliğindedir.
> Yazılım geliştiriciler SDL kullanarak birçok platform (Linux, Syllable, Haiku/BeOS, OpenVMS, Windows, Mac OS X, AmigaOS ve klonu MorphOS) için bilgisayar oyunları ve çoklu ortam uygulamaları geliştirebilirler.
> Zaman içerisinde C dilinin dışında C++, Perl, Python ve Pascal gibi birçok popüler dil içinde SDL kütüphaneleri geliştirilmiş, yaygın olarak kullanılmaktadır.

Yukarıda da yazdığı gibi kütüphanemiz C ile yazılmış, açık kaynak ve platform bağımsız bir kütüphanedir. Bu kütüphaneyi kullanarak 2D(2 Boyutlu) oyunlar yapabiliriz.

2 Sistemimize Kuralım

Bu kütüphane ayrıca sonradan yazılmış kütüphanelerle güçlendirilmiştir.Bu kütüphaneleride sistemimize kurarak kullanacağız.Bu kütüphaneleri Ubuntumuza kurmak için Synaptic Paket Yöneticisini açarak arama kısmına SDL yazalım.Gelen seçeneklerden aşağıdaki kütüphaneleri işaretleyelim ve kuralım.

```
1  libsdl-1.2debian
2  libsdl-1.2debian-alsa
3  libsdl-1.2dev
4  libsdl-image1.2
5  libsdl-image1.2-dev
6  libsdl-ttf2.0
7  libsdl-ttf2.0-dev
8  libsdl-mixer1.2
9  libsdl-mixer1.2-dev
10 libsdl-net1.2
11 libsdl-net1.2-dev
```

Bu paketleri kurmak şimdilik bizim için yeterli eğer gerekli olursa gerektiği zaman başka kütüphanelerde kurarız. Bu kütüphanelerden kurduğumuz ilk üç kütüphane SDL çekirdek kütüphaneleridir. İmage kütüphanesi grafik işlemleri ile alakalı kütüphanedir. SDL sadece .bmp uzantılı grafik dosyalarını hafızaya yükler ve işlem yapabilir.Diğer dosya formatlarını da kullanabilmek için bu kütüphaneden yararlanacağız.TTF(True Type Fonts) kütüphanesi ekrana belli fontlar kullanarak yazılar yazmak için kullanacağımız kütüphanedir. Sessiz bir oyun tabiki yeteri kadar heyacan vermiyecektir. Ses ile ilgili işlemler için Mixer kütüphanesini kullanacağız.Net kütüphanesinin işlevini ise tahmin edebilirsiniz, bu da network işlemleri ile alakalı kütüphanedir.

3 Derleme Ayarları

Kütüphanemizi sistemimize kurduktan sonra derleme işleminin nasıl yapılacağına göz atalım. Konsol kullanarak C++ kodlarını derlemek için

```
g++ kaynak.cpp -o program_adi
```

komutunu kullanırız. Eğer programımıza bir kütüphane bağlamak istersek kullanacağımız anahtar -l(küçük L, -1 değil), ardından da kütüphane ismi gelir.Bizimde yazdığımız oyun kodlarına SDL kütüphanesinin bağlanması gerekir. Yani vereceğimiz komut

```
g++ kaynak.cpp -o program_adi -lSDL
```

Çalıştırmak için

```
./program_adi
```

komutlarını kullanırız.Eğer bir IDE kullanıyorsak yapmamız gereken ayarlarda hemen hemen aynıdır.Burada örnek olarak NetBeans geliştirme ortamı için gerekli ayarları anlatacağım. Diğer geliştirme ortamları içinde ayarlar benzer olacaktır. Aslında yaptığımız iş derleyicinin Linker(Bağlayıcı) ayarlarını, parametrelerini ayarlamaktır. İlk olarak C++ projemizi açıyoruz. Sol üst tarafta açıtığımız projeyi görüyoruz. Bunun üzerine sağ tuşla tıklıyoruz ve Properties(Özellikler) kısmına giriyoruz. Burası ilgili proje üzerinde yapacağımız ayarların özellikleri bulunmaktadır, biz burada C++ derleyici ayarlarını yapacağımız için C++ compiler bölümüne giriyoruz. Açılan ayarlardan Command Line bölümü altındaki Additional Options bölümüne gerekli anahtarımızı giriyoruz -lSDL. Hepsi bu kadar.

4 SDL Alt Sistemleri

C++ programcıları bazı kütüphaneleri kullanırken ilk olarak ilklenmesi gerektiğini biliyorlardır. İşte SDL'de de aynı durum sözkonusudur. SDL fonksiyonları kullanılmadan önce SDL'nin ilklenmesi, çalıştırılması veya kullanıma hazırlanması gerekir. Artık siz bu işleme ne ad verirseniz.

SDL 8 tane alt sisteme sahiptir.Bunlar Ses, Video(Grafik), Timer(Zaman), CD-ROM, Olay Yakalama(Event), Çoklu Görev(Thread), Joystick ve Dosya Giriş-Çıkış sistemleridir.Bu sistemleri kullanmadan önce çalıştırılması gerekir.Bu çalıştırma programın başında olabileceği gibi gerektiği zamanda yapılabilir.Çalıştırmak için

```
1 SDL_Init() veya SDL_InitSubSystem()
```

komutları kullanılır. İkisi arasındaki fark SDL_Init() programın başında kullanılmalıdır. SDL_InitSubystem() ise çalışma anında istediğiniz alt sistemi çalıştırmamızı sağlar. Kullanım şekli, çalıştırılacak olan alt sistemin bayrakları(flag) argüman olarak bu komutlara geçilir. Örneğin Grafik(Video) alt sistemini çalıştırmak için

```
1 SDL_Init(SDL_INIT_VIDEO)
```

kodu kullanılır.Diğer alt sistemlerin bayrak listesi aşağıdaki gibidir.

```
1 SDL_INIT_EVERYTHING - Bütün sistemleri çalıştırır.
2 SDL_INIT_VIDEO - Grafik(Video)
3 SDL_INIT_TIMER - Zamanlayıcı
4 SDL_INIT_AUDIO - Ses
5 SDL_INIT_CDROM - CD-Rom
6 SDL_INIT_JOYSTICK - Joystick
7 SDL_INIT_EVENTTHREAD - Çok görevlilik(Thread)
8 SDL_INIT_NOPARACHUTE - SDL'in hata sinyallerini yakalamasını önler
```

Bu sistemleri işimiz bittiği zaman kapatmamız gerekir.Bunun için de

```
1 SDL_Quit()
```

komutu kullanılır.Herhangi bir argüman almaz, bütün açık olan alt sistemleri sonlandırır ve programı kapatır. Eğer biz sadece tek bir alt sistemi kapatmak istiyorsak

```
1 SDL_QuitSubSystem()
```

komutu kullanılır. Kapatmak istediğimiz alt sistemin bayrağını argüman olarak bildiririz.

Şimdi şunu sorabilirsiniz birden fazla alt sistemi nasıl çalıştırabiliriz? Cevabı basit çalıştırmak istediğimiz alt sistemlerin bayrak değişkenlerini bit düzeyinde işlem yapan veya bağlacı (|) ile bağlamamız gerekir.Mesela zamanlayıcı ve grafik alt sistemlerini çalıştırmak istersek

```
1 SDL_Init(SDL_INIT_TIMER | SDL_INIT_VIDEO)
```

kodunu yazmamız yeterlidir.

Son olarakta bir sistemin o anda çalışır vaziyette olup olmadığını kontrol etmek isteyebilirsiniz.Bunun içinde yapmanız gereken

SDL_WasInit() fonksiyonunu kullanmanız gerekir.Argüman olarak çalışıp çalışmadığı öğrenilmek istenilen alt sistemin bayrak değişkeni alır.Sıfırdan farklı bir değer dönüyorsa sistem çalışıyor durumdadır.Ufak bir kod parçası verecek olursak şöyle

```
1 if(SDL_WasInit(SDL_INIT_VIDEO)!=0)
2           printf("Video alt sistemi yüklü.\n");
3           else
4           printf("Video alt sistemi yüklü değil.\n");
```

Yukarıdaki kod zaten herşeyi anlatıyor.

5 Pencere Oluşturma

Sabırsızlandığınızı biliyorum ama işin temelini de göstermek gerektiğini düşünüyorum.Bu yüzden bu kadar hikayeyi yazdık.Şimdi ilk uygulamamızı yazacağız.Bu program sadece bir pencere oluşturuyor ve kapatıyor. İşte kodlar (Örnek 1)

```
1  //Ornek1-Pencere olusturma
2  #include <SDL/SDL.h>
3  #include <iostream>
4  using namespace std;
5  int main(){
6      if(SDL_Init(SDL_INIT_EVERYTHING)==-1){
7          cout<<"Butun sistemler baslatilamadi\n";
8          return 0;
9      }
10     cout<<"SDL sistemleri baslatildi.\n";
11     SDL_Surface *pencere=NULL;
12     pencere=SDL_SetVideoMode(640, 480, 32, SDL_SWSURFACE);
13     SDL_Flip(pencere);
14     SDL_Delay(3000);
15     SDL_Quit();
16     cout<<"SDL sistemleri durduruldu.\n";
17     return 0;
18 }
```

İlk 2 satırda bize gerekli olan başlık dosyalarını programımıza dahil ettik. İsim uzayımızı belirledik.Main fonksiyonumuzun içinden açıklamaya başlarsak, ilk olarak bütün SDL alt sistemlerini çalıştırdık bunu yaparken, if şartı ile kendimize hata yakalamak için kolaylık sağlayacak bazı tedbirler aldık. SDL_Init() fonksiyonu başarısız olursa -1 değeri döndürecek, ekrana hatanın nerde olduğunu yazacak ve program sonlandırılacak. Bu sayede programımızın neresinde sorun olduğunu anlayabileceğiz.Eğer çalıştırma işlemi başarılı olursa konsola başarılı olduğuna dair bir mesaj yazacak.Şimdi ilk defa göreceğiniz SDL_Surface(Türkçesi Yüzey) SDLde önceden tanımlanmış bir yapıdır.Yüzeyler bizim yüklediğimiz grafik dosyalarını tutmaya yarar.Burada pencere adlı bir yüzey işaretçisi tanımlıyoruz ve NULL değeri atıyoruz.Çünkü herhangi bir bellek sızıntısına imkan vermemek için, işaretçileri tanımladığımız anda onlara ilk değerlerini vermek aslında iyi bir yöntemdir. Daha sonra bu pencere adlı yüzeyimizi ekranda gösterilecek olan asıl yüzey olarak bazı özelliklerini ayarlıyoruz.Bunu SDL_SetVideo() fonksiyonu ile yapıyoruz.Bu fonksyonun parametreleri şu şekildedir.İlk iki parametre(örneğimizde 640,480) oluşturulacak pencerenin ekran çözünürlüğüdür. Üçüncü parametre(örneğimizde 32) ekranda bit başına düşen piksel sayısıdır.Son parametre ise pencernin özelliklerini belirten bir bayrak değişkeni alır.Örneğimizdeki bu bayrak değişkeni, pencere özelliklerinin sistem belleğinde tutulacağını söyler. Aşağıda diğer bayrak değişkenleride listelenmiştir.

Komut	Açıklama
SDL_SWSURFACE	Yüzeye ait bilgilerin sistem belleğinde tutulmasını sağlar.
SDL_HWSURFACE	Yüzeye ait bilgilerin ekran kartının belleğinde tutulmasını sağlar.

Komut	Açıklama
SDL_ASYNCBLIT	Asenkron yüzey göstermeyi aktif hale getirir. Bu genellikle tek işlemcili makinalarda bit işlemeyi (blit - bit block transfer - bit bloğu değişimi) yavaşlatır ama SMP sistemlerde hız artışı sağlayabilir.
SDL_ANYFORMAT	Normalde eğer video yüzeyi kullanılamayacak bir ekran derinliği (bpp) isterse SDL gölge bir yüzey ile bunu emule eder. SDL_ANYFORMAT bayrağı ile SDL'in bunu yapması engellenir ve SDL'in yüzeyin derinliğini umursamadan onu kullanması sağlanır.
SDL_HWPALETTE	SDL'e ayrıcalıklı palet erişimi verir. Bu bayrak olmadan SDL_SetColors komutu ile istediğiniz renge herzaman ulaşamayabilirsiniz.
SDL_DOUBLEBUF	Çifte tamponlamayı etkin hale getirir. Sadece SDL_HWSURFACE bayrağı ile beraber kullanılabilir.
SDL_FULLSCREEN	SDL tam ekran çalıştırmaya çalışıyor.
SDL_OPENGL	OpenGL render ekranı yaratır. SDL_GL_SetAttribute komutu ile OpenGL ayarlamalarına başlamadan önce bu bayrağın etkinleştirilmesi gerekir.

Komut	Açıklama
SDL_OPENGLBLIT	Üstteki gibidir ama aynı zamanda blitting (*yardım*) işlemlerine izin verir.
SDL_RESIZABLE	Boyutlandırılabilir bir pencere yaratır. Pencere boyutları değiştirildiği zaman SDL_VIDEORESIZE olayı tetiklenir ve SDL_SetVideoMode yeni boyut ile tekrar çağırılabilir.
SDL_NOFRAME	Mümkün ise çerçevesiz bir pencere yaratır. Tam ekran modu otomatik olarak bu bayrağı etkinleştirir.

Bir video modunun uygun olup olmadığınıda kontrol edebilirsiniz.Bunun için kullanmanız gereken,

```
1 SDL_VideoModeOK(640,480,32,SDL_SWSURFACE)
```

eğer uygun değil ise false, uygunsa true değerleri döner.Örnek bir kullanım ise şöyle verilebilir.

```
1     if (!SDL_VideoModeOK(640, 480, 32, SDL_HWSURFACE))
2             cout<<"Ekran modu uygun değil.\n";
3             else
4             cout<<"Ekran modu uygun.\n";
```

SDL_Flip() fonksiyonu argüman olarak bildirilen yüzeyi günceller.Burada pencere adlı yüzeyi ekranda gösterilecek olan asıl yüzey olarak ayarlamıştık, bu ayarların güncellenmesi gerekir ki bu yüzey ekranda gözüksün. Bu işlemi SDL_Flip() fonksiyonu ile yapıyoruz. Bu fonksiyonu kaldırıp tekrar derleyerek programınızdaki değişiklikleri gözlemleyebilirsiniz.

SDL_Delay() fonksiyonu penceremizin hemen kapanmaması için kullandığımız bir fonksiyondur.Argüman olarak verilen (milisaniye cinsinden) süre kadar programı bekletir. İleride programı bir ana döngü içerisine aldığımız zaman buna ihtiyacımız kalmayacaktır.Programın çıktısı aşağıdaki gibi olacaktır.

6 Merhaba Dünya

Pencere açmayı öğrendiğimize göre bu pencereye bir grafik koymakla işe başlayalım. Bu işlem hakkında ön bilgi vermek gerekirse bir kaç söz söyleyebiliriz. İlk olarak grafik dosyası hafızaya yüklenir bu yüklenen dosya tabiki SDL_Surface yapısı kullanılarak hafızada tutulur. Yüklediğimiz grafiğin tutulduğu yüzeyi ana pencere olarak kullanacağımız yüzeyin üstüne uyguluyoruz. Uygulamakla grafik hemen gözükmez, gözükmesi için bu yüzeyin yenilenmesi gerekir. Yüzeyimizi yeniledikten sonra grafiğimiz artık ekranda gözükecektir.Yuklemek istediğimiz grafik dosyası programımız ile aynı dizinde olmalıdır.Fonksiyona argüman olarak dosyanın ismi verilir, eğer farklı bir yerde ise dosyanın dizinide yazılmalıdır.

```
1  //ornek2- grafik yuklemek
2  #include <SDL/SDL.h>
3  #include <iostream>
4  using namespace std;
5  int main(){
6      if(SDL_Init(SDL_INIT_EVERYTHING)==-1){
7          cout<<"Butun sistemler baslatilamadi\n";
8          return 0;
9      }
10     cout<<"SDL sistemleri baslatildi.\n";
11     SDL_Surface *pencere=NULL;
12     SDL_Surface *grafik=NULL;
13     pencere=SDL_SetVideoMode(640, 480, 32, SDL_SWSURFACE);
14     grafik=SDL_LoadBMP("merhaba.bmp");
15     SDL_BlitSurface(grafik,NULL,pencere,NULL);
16     SDL_Flip(pencere);
17     SDL_Delay(3000);
18     SDL_FreeSurface(grafik);
19     SDL_Quit();
20     cout<<"SDL sistemleri durduruldu.\n";
21     return 0;
22 }
```

Programda ilk olarak iki tane Surface tipinde işaretçi tanımlıyoruz. Bunlardan birisini ana pencere olarak, diğerini ise yüklediğimiz grafiği tutmak için kullanacağız. Grafiği yüklemek için SDL_LoadBMP(); fonksiyonunu kullanıyoruz.Bu SDL varsayılan grafik yükleme fonksiyonudur, sadece .bmp uzantılı dosyaları yükler.Merak etmeyin başka uzantılı grafik dosyalarını yüklemeyide öğreneceğiz ve bunlarda oldukça basit, image kütüphanesini boşuna yüklemedik.Merhaba.bmp isimli grafik dosyasını hafızaya yükledik ve yerini grafik adlı işaretçi ile tutuyoruz. SDL_BlitSurface() fonksiyonu bir yüzeyi bir başka yüzey üzerine uygular. Bunu iki tabakayı üst üste yapıştırmak gibi de düşünebilirsiniz.Argüman sırası üste uygulanacak olan yüzey, ve bunun gerekli ayarları, üstüne uygulanacak olan yüzey, ve bu yüzeyin ayarları. Şimdilik ayarları NULL yani boş geçiyoruz.İlerde bunlarla alakalı örneklerimiz olacak.Burada tahmin ettiğiniz gibi ilk yüzey ikinci yüzey üzerine uygulanıyor.SDL_Flip(pencere) fonksiyonu ile pencere yüzeyi üzerine bir başka yüzey uyguladığımız için bu yüzeyin tekrar güncellenmesi gerekiyor.Böylece iki yüzey tek bir yüzey halini alıyor.Grafik adlı yüzey ile işimiz bittiğine göre bu bellek alanını serbest bırabiliriz, bunu da SDL_FreeSurface() ile yapıyoruz.Programın çıktısı aşağıdaki gibi olmalıdır.

Şimdilik bu kadar yeterli. Bir sonraki yazıda grafik işlemlerine devam edeceğiz.Grafiğimizi pencere üzerinde istediğimiz yere yerleştirme, kesme gibi işlemleri öğreneceğiz. Yazdığımız örneklerde kapatma tuşu çalışmayacaktır. Bunun için endişelenmeyin, ilerleyen konularda(Events, Olay yakalama) bu sorunu çözeceğiz.

7 Yazı içerisindeki dosyalar

Dosyaları İndir