SDL ile Oyun Programcılığı - II

Sinan Ateş

Nisan, 2016

İçindekiler

Bir önceki yazımızda SDL hakkında biraz bilgi vermiştik. Uygulama olarakta SDL alt sistemlerini çalıştırmayı, bir pencere oluşturmayı, belleğe grafik dosyasını yüklemeyi ve bunu ekranda göstermeyi yapmıştık. Şimdi kaldığımız yerden devam ediyoruz.

Grafik dosyamızın .bmp uzantılı olması gerekiyordu ama biz başka bir uzantılı dosya ile çalışmak istiyorsak bunun için SDL_image kütüphanesini kurmuş olmamız gerekiyor. Eğer bu kütüphaneyide kurduysak yapmamız gereken tek şey grafik dosyamızı yüklemek için kullandığımız fonksiyon olan SDL_LoadBMP() fonksiyonu yerine IMG_Load() fonksiyonunu kullanmak. Bu fonksiyon bir çok dosya uzantısını desteklemektedir. Ayrıca yüklediğimiz grafik dosyasının renk derinliğinden hiç bahsetmedik. Acaba yüklediğimiz dosya ile oluşturduğumuz pencerenin renk derinliği aynı mı? Bunun içinde endişelenmeye gerek yok, SDL bize bu ayarları yapmamız için bir hazır fonksiyon sunuyor. Bu fonksiyonumuzda SDL_DisplayFormat(SDL_Surface*) dir. Argüman olarak geçtiğimiz yüzey üzerindeki grafik dosyamızda gerekli ayarlamaları yaptıktan sonra geriye bir SDL_Surface tipinde bir işaretçi döndürür. Artık bu fonksiyonları kullanarak kendi grafik yükleme fonksiyonumuzu yazabiliriz.İşte bizim fonksiyonumuz

```
1  SDL_Surface *resimYukle(std::string dosyaAdi){
2      SDL_Surface *yuklenenResim=NULL;
3      SDL_Surface *optimizeResim=NULL;
4      yuklenenResim=IMG_Load(dosyaAdi.c_str());
5      if(yuklenenResim != NULL){
6          cout<<dosyaAdi<<" adli dosya basariyla yuklendi.\n";
7          optimizeResim=SDL_DisplayFormat(yuklenenResim);
8      }
9      SDL_FreeSurface(yuklenenResim);
10     return optimizeResim;
11 }
```

Argüman olarak dosyamızın ismini veriyoruz. Fonksiyon içerisinde iki tane yüzey tanımladık. Bunlardan birisi dosyamızın ilk halini diğeri ise bütün ayarlamaların yapıldıktan sonraki halini tutacaktır. yuklenenResim adlı yüzey değişkenimize dosyamızı yüklüyoruz.Eğer işlem başarılı olursa konsola bir mesaj yazacak ve yüzey gerekli ayarlamaların yapılması için SDL_DisplayFormat() fonksiyonuna gönderilecektir. Buradan ayarlamaların yapılmış halini ise optimizeResim adlı değişken ile alıyoruz.Eski hali ile işimiz bittiğine göre bu yüzeyi serbest bırakabiliriz. optimizeResim adresinide ana programımıza geri döndürüyoruz. Bir hatırlatma olarak IMG_Load() fonksiyonunun kullanılabilmesi için derleme parametrelerine -lSDL_image şeklinde image kütüphanesininde bağlanması gerektiğini söyleyelim.

Grafik dosyamızı pencere üzerinde istediğimiz bir konuma yerleştirmek için bir SDL yapısı olan SDL_Rect yapısından faydalanacağız. SDL_Rect bir dikdörtgen yapısıdır ve SDL içerisinde önceden tanımlanmıştır. Tanımlanışı ve içerisinde tanımlı değişkenler şu şekildedir.

```
1  typedef struct{
2              Sint16 x, y;
3               Uint16 w, h;
4          } SDL_Rect;
```

Gördüğünüz gibi içerisinde dört tane değişken var.Bunlardan x ve y değişkenleri dikdörtgenimizin sol üst köşesinin koordinatlarıdır. w değişkeni dikdörtgenimizin genişliği, h ise yüksekliğidir. Bu değişkenler bizim bir bölgeyi SDL'ye bildirebilmemiz için yeterlidir. Koordinatlar demişken SDL nin koordinatları nasıl tanımladığınada bakalım. Penceremizin sol üst köşesi (0,0) başlangıç noktası (orijin) kabul edilir.Sağa doğru x değişkeni artarken aşağı doğruda y değişkeni artar. Görüldüğü gibi normal Kartezyen koordinat sisteminden tek farkı y değişkeninin yukarı değilde aşağıya doğru artmasıdır.

Buraya kadar verdiğimiz bilgilerin uygulaması olarak bir örnek yapalım.Örnek programımızda 800-600 boyutunda bir pencere oluşturuyoruz, fakat arkaplan olarak kullanmak istediğimiz grafiğin boyutu 400-300 piksel boyutunda, bizde bu grafik dosyasını ekranın dört köşesine yerleştirerek kullanmak istiyoruz.İşte bunun için arkaplanımızın nereye yerleşeceğini bildirmemiz için SDL_Rect tipinde yapistirilacakYer adlı birdeğişken kullanıyoruz.Bu sayede istediğimiz bir bölgeyi SDL'ye bildirebiliyoruz.Sadece sol üst köşenin koordinatlarını bildirmemiz yeterli çünkü SDL arkaplanımızın genişliğini ve yüksekliğini bildiği için diğer bilgileri varsayılan olarak doldurur.

Arkaplanlarımızı yerleştirdikten sonra üstüne bir grafik dosyamızı da yerleştirmek istiyoruz. Fakat grafiğimizin üzerinde bir Linux yazısı ve bir Tux resmi var.Bizim istediğimiz sadece Linux yazısını almak, bunun içinde kesilecekYer adlı bir SDL_Rect değişkeni kullanıyoruz.Linux yazısının bulunduğu bölgenin sol üst köşe koordinatları (0,45) genişliği 150, yüksekliği ise 90 pikseldir.Bütün bu ayarları kesilecekYer adlı yapı ile SDL'ye bildiriyoruz. Bu ayarları SDL içerisinde nasıl kullandığımıza gelirsek iki yüzeyi birleştirme, üst üste yapıştırma fonksiyonumuz olan SDL_BlitSurface() fonksiyonuna geri dönelim.Argümanlarından iki tanesini biliyorduk.Birincisi üste yapıştırılacak olan yüzey, üçüncüsü ise hangi yüzeyin üzerine yapıştırılacağını bildiren argümanlardı. Bu argümanlardan ikinci ve dördüncü argüman da kendinden bir önceki yüzeyin bölgelerini belirler. Mesela ikinci argümanda bildirilen bölge birinci yüzey üzerinden kesilir ve üçüncü argümanda bildirilen yüzeye dördüncü argümanda bildirilen bölgesine yapıştırılır. Galiba biraz karışık oldu. :) Kodlarla konuşmak daha iyi, işte ilgili kod parçası

```
1 SDL_BlitSurface(grafik,&kesilecekYer,pencere,&yapistirilacakYer);
```

Dikkat ederseniz bölgeleri argüman olarak geçerken, değişkenlerin adresleri veriliyor. Bu yüzden & işareti ile birlikte yazılıyor. İşte bütün kodlar ve ekran çıktısı:

```
1  //ornek3.cpp
2  #include <SDL/SDL.h>
3  #include <SDL/SDL_image.h>
4  #include <string>
5  #include <iostream>
6  using namespace std;
7
8  SDL_Surface *pencere=NULL;
9  SDL_Surface *arkaplan=NULL;
10 SDL_Surface *grafik=NULL;
11
12 SDL_Rect kesilecekYer;
13 SDL_Rect yapistirilacakYer;
14
15 SDL_Surface *resimYukle(std::string dosyaAdi){
16     SDL_Surface *yuklenenResim=NULL;
17     SDL_Surface *optimizeResim=NULL;
18     yuklenenResim=IMG_Load(dosyaAdi.c_str());
19     if(yuklenenResim != NULL){
20         cout<<dosyaAdi<<" adli dosya basariyla yuklendi.\n";
21         optimizeResim=SDL_DisplayFormat(yuklenenResim);
22     }
23     SDL_FreeSurface(yuklenenResim);
24     return optimizeResim;
25 }
26
27 int main(){
28     if(SDL_Init(SDL_INIT_EVERYTHING)==-1){
29         cout<<"Butun sistemler baslatilamadi\n";
30         return 0;
31     }
```

```
32      pencere=SDL_SetVideoMode(800, 600, 32, SDL_SWSURFACE);
33
34      arkaplan=resimYukle("arkaplan.png");  //Grafikler bellege yuklendi ve
35      grafik=resimYukle("grafik.jpeg");          // gerekli ayarlamalar yapildi
36
37      yapistirilacakYer.x=0;
38      yapistirilacakYer.y=0;
39      SDL_BlitSurface(arkaplan,NULL,pencere,&yapistirilacakYer);  //Sol üst kose
40
41      yapistirilacakYer.x=400;
42      yapistirilacakYer.y=0;
43      SDL_BlitSurface(arkaplan,NULL,pencere,&yapistirilacakYer);  //Sag üst kose
44
45      yapistirilacakYer.x=0;
46      yapistirilacakYer.y=300;
47      SDL_BlitSurface(arkaplan,NULL,pencere,&yapistirilacakYer);  //Sol alt kose
48
49      yapistirilacakYer.x=400;
50      yapistirilacakYer.y=300;
51      SDL_BlitSurface(arkaplan,NULL,pencere,&yapistirilacakYer);  //Sag alt kose
52
53      kesilecekYer.x=0;
54      kesilecekYer.y=45;
55      kesilecekYer.w=150;
56      kesilecekYer.h=90;
57      yapistirilacakYer.x=325;
58      yapistirilacakYer.y=255;
59      SDL_BlitSurface(grafik,&kesilecekYer,pencere,&yapistirilacakYer);
60
61      if(SDL_Flip(pencere)==-1){
62          cout<<"Ekran Guncellenemedi\n";
63          return 1;
64      }
65
66      SDL_Delay(4000);
67      SDL_FreeSurface(arkaplan);
68      SDL_FreeSurface(grafik);
69      SDL_Quit();
70  }
```

Ekrana grafikleri yerleştirebildiğimize göre artık ekrana yazılı mesaj yazmayıda inceleyebiliriz. Bu işlem için SDL_ttf True Type Fonts kütüphanesini kullanacağız. Bu kütüphanede diğer kütüphaneler gibi ilklendirilmesi ve kullanıma hazırlanması gerekir. İçerisinde font dosyalarını açmaya, kullanmaya yarayan fonksiyonlar ve değişkenler bulunur. Sıradaki örneğimiz bu kütüphanenin kullanımını basitçe göstermektedir. Kullanacağımız font dosyasıda programımızla aynı dizinde olmalı ya da yeri bildirilmelidir.

Bu programımızda grafikleri belirttiğimiz konuma yerleştiren bir fonksiyon yazdık.Bu sayede kod tekrarından kurtulacağız.Bir önceki örnekten hatırlarsanız bir çok kod tekrarı vardı.

Programı derlemek için derleme parametrelerine -lSDL -lSDL_image -lSDL_ttf eklemeyi unutmayın. Konsoldan derlemek icin:

```
1  g++ ornek4.cpp -o ornek4 -lSDL -lSDL_image -lSDL_ttf
2  //ornek4.cpp
3  #include <SDL/SDL.h>
4  #include <SDL/SDL_image.h>
5  #include <SDL/SDL_ttf.h>
6  #include <string>
7  #include <iostream>
8  using namespace std;
9
10 SDL_Surface *pencere=NULL;
11 SDL_Surface *arkaplan=NULL;
12 SDL_Surface *yazi=NULL;
```

```
13
14 TTF_Font *font; //kullanacagimiz font dosyasina bir isaretci
15 SDL_Colour yaziRengi = { 255, 255, 255 }; //yazi rengi
16
17 SDL_Surface *resimYukle(std::string dosyaAdi){
18     SDL_Surface *yuklenenResim=NULL;
19     SDL_Surface *optimizeResim=NULL;
20     yuklenenResim=IMG_Load(dosyaAdi.c_str());
21     if(yuklenenResim != NULL){
22         cout<<dosyaAdi<<" adli dosya basariyla yuklendi.\n";
23         optimizeResim=SDL_DisplayFormat(yuklenenResim);
24     }
25     SDL_FreeSurface(yuklenenResim);
26     return optimizeResim;
27 }
28 void yuzeyeUygula(int x,int y,SDL_Surface *kaynak,SDL_Surface *hedef){
29     SDL_Rect bolge;
30     bolge.x=x;
31     bolge.y=y;
32     SDL_BlitSurface(kaynak,NULL,hedef,&bolge);
33 }
34 int main(){
35     if(SDL_Init(SDL_INIT_EVERYTHING)==-1){
36         cout<<"Butun sistemler baslatilamadi\n";
37         return 0;
38     }
39     if(TTF_Init()==-1){
40         cout<<"Font kütüphanesi calistirilamadi.\n";
41         return 1;
42     }
43     pencere=SDL_SetVideoMode(800, 600, 32, SDL_SWSURFACE);
44
45     arkaplan=resimYukle("arkaplan2.png");
46     yuzeyeUygula(0,0,arkaplan,pencere);
47
48     font = TTF_OpenFont( "FreeMono.ttf", 30 );      //Font dosyamizi actik, ikinci arguman
           yazi buyuklugu(punto)
49     yazi = TTF_RenderText_Solid( font, "Merhaba SUDO okurlari", yaziRengi );    //Yazimizi
           yuzeye uyguladık
50     yuzeyeUygula(20,20,yazi,pencere);           //Keskin kenarlı gecisler sert
51
52     yazi = TTF_RenderText_Blended( font, "Merhaba SUDO okurlari", yaziRengi );
53     yuzeyeUygula(20,70,yazi,pencere);           //Renk gecisleri yumusatilmis
54
55     SDL_Colour dolguRengi={255,0,0};   //Kirmizi
56     yazi = TTF_RenderText_Shaded(font,"Merhaba SUDO okurlari",yaziRengi,dolguRengi);
57     yuzeyeUygula(20,120,yazi,pencere);          //Yazi dolgulu olarak yaz
58
59     TTF_SetFontStyle(font, TTF_STYLE_BOLD );        //Fontu kalin olarak bicimlendirdik
60     yazi = TTF_RenderUTF8_Solid( font, "Merhaba SUDO okurlari", yaziRengi );
61     yuzeyeUygula(20,170,yazi,pencere);
62
63     TTF_SetFontStyle(font, TTF_STYLE_ITALIC );      //Fontu italik olarak bicimlendirdik
64     yazi = TTF_RenderUTF8_Solid( font, "Merhaba SUDO okurlari", yaziRengi );
65     yuzeyeUygula(20,220,yazi,pencere);
66
67     TTF_SetFontStyle(font, TTF_STYLE_UNDERLINE );       //Fontu altı cizili olarak
           bicimlendirdik
68     yazi = TTF_RenderUTF8_Solid( font, "Merhaba SUDO okurlari", yaziRengi );
69     yuzeyeUygula(20,270,yazi,pencere);
70
71     TTF_SetFontStyle(font, TTF_STYLE_ITALIC | TTF_STYLE_UNDERLINE | TTF_STYLE_BOLD );
           //Fontu kalin olarak bicimlendirdik
72     yazi = TTF_RenderUTF8_Solid( font, "Merhaba SUDO okurlari", yaziRengi );
73     yuzeyeUygula(20,320,yazi,pencere);
74
```

```
75      TTF_SetFontStyle(font, TTF_STYLE_NORMAL);
76      yazi = TTF_RenderUTF8_Solid( font, "Merhaba SUDO okurlari", yaziRengi );
77      yuzeyeUygula(20,370,yazi,pencere);
78
79      if(SDL_Flip(pencere)==-1){
80          cout<<"Ekran Guncellenemedi\n";
81          return 1;
82      }
83
84      SDL_Delay(4000);
85      SDL_FreeSurface(arkaplan);
86      SDL_FreeSurface(yazi);
87      TTF_CloseFont( font );
88      TTF_Quit();
89      SDL_Quit();
90  }
```

Örnek programımızda font dosyasını açıyoruz ve SDL_Font tipindeki bir işaretçi ile kullanacağımız font bilgilerini alıyoruz. Ayrıca yazımızın punto değerini de burada belirledik.

```
1  TTF_RenderText_Solid( font, "Merhaba SUDO okurlari", yaziRengi )
```

Bu fonksiyon yardımıyla font adlı işaretçi ile bildirdiğimiz font ve yazıRengi adlı değişkenler ile yazımız bir yüzey üzerine uygulanıyor.Bundan sonra bu yüzeyi herhangi bir grafik dosyası gibi kullanabiliriz. Burada argüman sıralarını görüyorsunuz. Bu fonksiyonun

```
1  SDL_Surface *TTF_RenderGlyph_Solid(TTF_Font* font, Uint16 ch, SDL_Color fg);
2  SDL_Surface *TTF_RenderText_Solid(TTF_Font *font, const char *text, SDL_Color fg);
3  SDL_Surface *TTF_RenderUTF8_Solid(TTF_Font *font, const char *text, SDL_Color fg);
4  SDL_Surface *TTF_RenderUNICODE_Solid(TTF_Font *font, const Uint16 *text, SDL_Color fg);
```

gibi çeşitleride mevcuttur .Ayrıca aynı fonksiyonun örnektede gördüğünüz gibi Shaded ve Blended tipleride mevcuttur.Bunlar sırasıyla dolgulu yazma ve yumuşak renk geçişi kullanarak yazmayı sağlar. Shaded fonksiyonu diğerlerinden bir fazla argüman alır.Bu da dolgu rengidir. Renk ile ilgili bilmemiz gerekenler ise SDL_Color tipindeki bir yapı ile renk değerlerini saklayabiliriz. Bir rengi 3 değer ile temsil edebiliriz.Bunlar sırasıyla Kırmızı, Yeşil ve Mavidir. Bu sistem İngilizce baş harflarının bir araya gelmesiyle Red, Green, Blue RGB olarak adlandırılır.Ayrıca bunlara ilave olarak saydamlık ölçüsü eklenirse yani Alpha, sistem RGBA olarak adlandırılır. Üç sayı değerimiz 0 ile 255 arasında değer alabilir. 0 olması o renkten hiç katılmaması, 255 olması ise o renkten maksimum miktarda katılması demektir. Örneğin beyaz bütün renklerin maksimum değerde karışımıyla elde edilir(255,255,255). Siyah hiç bir rengin olmaması durumudur(0,0,0). Kırmızı elde etmek için kırmızıdan maksimum miktarda karıştırılması gerekir (255,0,0)

Yazı olurda biçimlendirme olmaz mı?

Örneğimiz de gördüğünüz gibi TTF_SetFontStyle(font, TTF_STYLE_UNDERLINE) fonksiyonu ile fontumuzu biçimlendirdik.Burada altı çizili olarak ayarladık.Bundan sonra yeniden ayarlayana kadar altı çizili yazacaktır. Tekrar eski haline getirmek için TTF_STYLE_NORMAL kullanılır. İşte programımızın çıktısı :

Buradaki resimde bütün ayrıntılar gözükmeyebilir. Kendi yazıp çalıştırdığınız programda ayrıntıları göreceksiniz.

Bu aylıkta burada bırakalım. Bir sonraki yazıda artık Events(olay) bir başka deyişle olay yakalamaya geçeceğiz. Bu sayede klavye ve mouseumuzu da işin içine sokacağız. Bunlarıda öğrendikten sonra vaktimiz kalırsa hep beraber bir oyun yazabiliriz.

Şimdilik Hoşçakalın….

0.1 Yazı içerisindeki dosyalar

Dosyaları İndir