Ankara Üniversitesi
Nallıhan Meslek Yüksekokulu

KOD PARÇACIKLARI

NBP108 - GRAFİK ANİMASYON II

Öğr.Gör. Salih ERDURUCAN

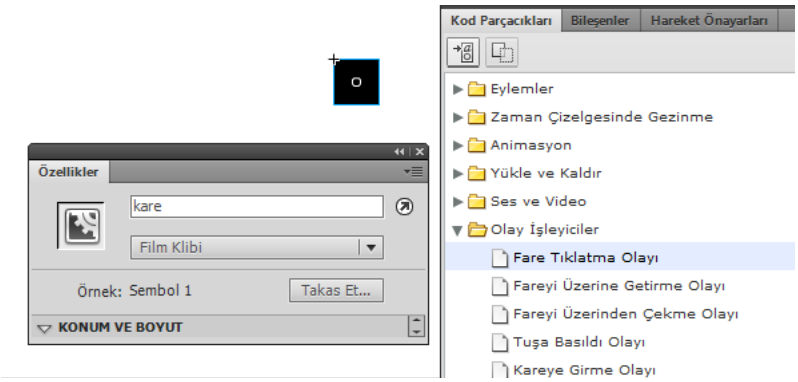
Kod Parçacıkları (CodeSnipet)

Kod parçacıkları paneli animasyon yazılımının beşinci versiyonunda gelmiş bir yeniliktir. Kod yazmak yerine panelden ilgili kodu seçip çift tıklamak yeterlidir. Panel; eylemler, zaman çizgisi kontrolleri, hazır animasyonlar, dışarıdan dosya yükle ve kaldır, ses ve video, olay yöneticileri şeklinde altı bölümden oluşmaktadır.

Paneli açmak için Pencere > Kod parçacıkları komutu ya da ActionScript editör penceresinin köşesinden seçilip açılabilir. Kodların sahneye düzgün eklenip çalışması için sahneye o kodun atanacağı bir nesne olmak zorundadır. Nesnenin örnek isminin (instanse name) tanımlanmış olması gereklidir yoksa panel kendisi o nesneye bir isim verecektir.

Kod parçacıklarını kullanmakiçin:

- Sahnedeki kare nesnesine seçin ve kod parçacıkları bölümünü açın.



- İstediğiniz eylemi seçerek çift tıklayın. Kod otomatik olarak eklenecektir.

```
/* Fare Tıklatma Olayı
Belirtilen sembol örneğini tıklattığınızda
kendi özel kodunuzu ekleyebileceğiniz bir işlev yürütülür.
Talimatlar:
1. Aşağıda "// Özel kodunuzu başlatın" yazan satırdan sonra gelen
yeni bir satıra özel kodunuzu ekleyin.Sembol örneği tıklatıldığında
kod yürütülür.
*/
kare.addEventListener(MouseEvent.CLICK, fl_MouseClickHandler_3);

function fl_MouseClickHandler_3(event:MouseEvent):void
{
    // Özel kodunuzu başlatın
    // Bu örnek kod, Çıktı panelinde "Fare Tıklatıldı"
    //sözcüklerini görüntüler.
    trace("Fare tıklatıldı");
    // Özel kodunuzu sonlandırın
}
```

ACTIONSCRİPT 3.0'de Zamana Bağlı Çalışan ve Tekrar Eden Olaylar

ActionScript 3.0 içerisinde ENTER_FRAME, TIMER ve TIMER_COMPLETE olayları en fazla kullanılan olaylar arasındadır. ENTER_FRAME olayı uygulama başladığından itibaren sonuna kadar çalışır. Timer olaylarında çalışma sürelerini belirlenebilir.

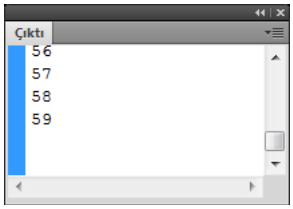
ENTER_FRAME

Bu olay sahne çalışma hızına bağlı olarak (Frame Rate) sürekli fonksiyonları çalıştırır. Sahne çalıştırıldığı anda tetiklenmeye başlar ve sahne hızına göre sürekli olarak uygulama kapatılana kadar çalışır.

Sahne çalışma hızı değiştirildiğinde tetiklenen fonksiyonun çalışması da değişecektir. Uygulama süresince tetiklenen fonksiyon çalışacağından işi bittiğinde tetikleyiciyi silmek doğru bir davranış olacaktır.

```
1  var say:uint=0;
2  stage.addEventListener(Event.ENTER_FRAME,calis);
3
4  function calis(event:Event):void
5  {
6  trace(say++);
7  }
```

Yukarıdaki sahneyi izleyen tetikleyici çalıştığı anda *calis* fonksiyonu tetiklenecek ve fonksiyon içindeki *say* değişkeni, sahne hızına göre kendini arttıracaktır. Çıkış ekranında say değişkeninin değerinin arttığını görülebilir. Sahne hızı arttırıldığında *say* değişkenin değeri daha hızlı artacaktır.



TİMER ÖRNEĞİ

ActionScript 3.0'da zamanlama işlevlerini işlemenin tercih edilen yolu, bir aralığa her ulaşıldığında olayları göndermek için kullanılabilen Timer sınıfının (flash.utils.Timer) kullanılmasıdır.

Bir zamanlayıcıyı başlatmak için, ilk olarak bir Timer sınıfı örneğini oluşturarak bu sınıfa ne sıklıkla bir zamanlayıcı olayı oluşturacağını ve durmadan önce kaç defa bunu yapacağını bildirirdiniz.

Örneğin, aşağıdaki kod, her saniye bir olay gönderen ve 60 saniye boyunca bu işleme devam eden bir Timer örneğini oluşturur:

var oneMinuteTimer:Timer = new Timer(1000, 60);

Timer nesnesi, belirli bir aralığa her ulaşıldığında bir TimerEvent nesnesi gönderir. TimerEvent nesnesinin olay türü timer (TimerEvent.TIMER sabitiyle tanımlı) olur. TimerEvent nesnesi, standart bir Event nesnesiyle aynı özellikleri içerir.

Timer örneği sabit sayıda bir aralığa ayarlanırsa, son aralığa ulaştığında bir timerComplete olayı da (TimerEvent.TIMER_COMPLETE sabitiyle tanımlı) gönderir.

Örnek: İlk olarak flash programını açalım ve sahneye **2 adet metin aracı** ekleyelim. Metin aracının birine **Saniye:** yazdım diğerine de **0** (sıfır) yazdım ve bu metin aracını **dinamik** metne dönüştürüp, örnek adını **say** yaptım.

Zaman çizelgesinde **1.kareye** tıklayıp **Eylemler** penceresini açalım ve aşağıdaki kodları yazalım.

```
1   import flash.utils.Timer;
2   import flash.events.TimerEvent;

3   var vakit:Timer=new Timer(1000);
4   vakit.start();

5   function zamanlama(event:TimerEvent):void{
6     say.text=String(int(say.text)+1);
7   }
8   vakit.addEventListener(TimerEvent.TIMER,zamanlama);
```

Kodları açıklayacak olursak;

- **1. ve 2.satırda,** Timer kütüphaneleri ekleniyor.
- **3.satırda, vakit** adında **Timer** nesnesi oluşturuluyor. Timer(1000) ile Timer'ın **çalışma aralığını** belirliyoruz. **1000 = 1 saniye** demektir. Yani timer nesnesi her 1 saniyede çalışacaktır. Burayı 10 saniyeye ayarlayacaksanız 10000 yazmalısınız.
- **4.satırda,** Timer nesnesini başlatıyoruz. Yani her 1 saniyede timer çalışacak. **Durdurmak** isterseniz **vakit.stop();** yazmalısınız.
- **5,6,7.satırlarda, zamanlama** adında fonksiyon tanımlanıyor.
- **6.satırda, say** adındaki metin aracının değerini 1 arttırıyoruz. İlk başta 0 girmiştik.
- **8.satırda,** burada da **vakit** Timer nesnesine olay ekliyoruz. Timer her çalışmasında **zamanlama** fonksiyonunu çalıştıracak. Kodları yazdıktan sonra **Ctrl+Enter** ile çalıştırabilirsiniz

Kaynak .

1- Animasyon Temelleri - MEGEP Eğitim Modülleri Erişim Tarihi : 20.02.2018
Url : http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf/Animasyon%20Temelleri.pdf