

MACROMEDIA FLASH. ОСНОВЫ ПРОГРАММИРОВАНИЯ. ПРОСТЕЙШИЕ СКРИПТЫ. УРОК 3

ПОЛУЧЕНИЕ СВОЙСТВ МС

Откройте новый файл и, используя инструмент **Text Tool**, создайте текстовое поле (рисунок 1).

Проследите за тем, чтобы в опциях текстового поля стояло либо **Input Text**, либо **Dynamic Text** (рисунок 2). В данном примере можно воспользоваться любым из вариантов.

Далее необходимо выбрать инструмент **Selection Tool** и размножить текстовое поле в восьми экземплярах (рисунок 3).

В панели **Properties** в поле **Var** задайте имена переменных для каждого из полей. Пусть это будут следующие имена переменных: **x, y, xsc, ysc, alpha, rotate, w, h**. Теперь создадим МС, свойства которого и будем получать. Для этого, как и ра-

нее, выберите инструмент **Rectangle**, при помощи которого нарисуйте на сцене квадрат (рисунок 4), выберите инструмент **Selection Tool** и выделите квадрат (рисунок 5), после чего преобразуйте его в кнопку (F8, выбрать **Button**). Раскройте панель **Actions** и в ней напишите следующий скрипт:

```
on (release) {  
    _root.x=this._x;  
    _root.y=this._y;  
    _root.xs=this._xscale;  
    _root.ys=this._yscale;  
    _root.w=this._width;  
    _root.h=this._height;  
    _root.alpha=this._alpha;  
    _root.rotate=this._rotation;  
}
```



Рисунок 4.



Рисунок 5.



Далее необходимо ... размножить текстовое поле в восьми экземплярах.

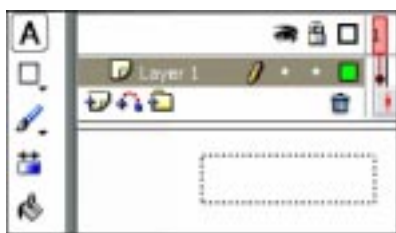


Рисунок 1.

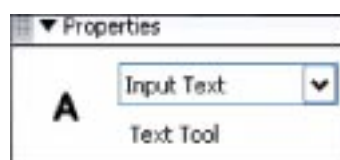


Рисунок 2.



Рисунок 3.

После этого еще раз выберите кнопку и преобразуйте ее в МС (F8, **Movie Clip**). Таким образом, получилась следующая конструкция: кнопка со скриптом, находящаяся внутри МС. Теперь осталось запустить **Control** → **Test movie** и посмотреть, что же получилось. По нажатию ЛКМ на квадрат, в текстовых полях отображаются свойства МС. Теперь остается «размножить» МС по сцене, изменить их параметры (угол поворота, прозрачность, координаты, масштабы) и снова запустить на просмотр. При выборе того или иного МС в текстовых полях будут возникать значения свойств МС (рисунок 6).

Все, пример выполнен. Хотелось бы отметить, что координаты и другие свойства отображаются не в виде целого, а в виде вещественного числа – это одно из свойств МС, с которым нужно примириться и учитывать при дальнейшей работе (особенно с тем, что у МС могут быть координаты с дробной частью).

ЗАДАНИЕ СВОЙСТВ МС

Во Flash существует возможность не только получать свойства МС, но и задавать их самостоятельно. На это и рассчитан следующий пример.

Откройте новый файл и создайте шесть текстовых полей с атрибутом **Input Text**. Если в предыдущем примере можно было выбрать **Input Text** или **Dynamic Text**, то здесь нужно будет вводить значения в текстовые поля, и, следовательно, **Input Text** должно быть обязательно. Задайте имена

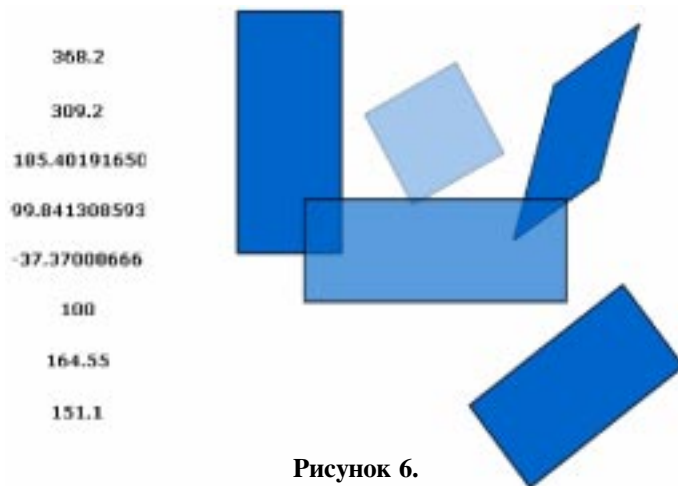


Рисунок 6.

переменных, которые будут вводиться в эти текстовые поля: **x**, **y**, **xs**, **ys**, **rotate**, **alpha**. Поскольку данные должны быть введены в поля, то желательно, чтобы эти текстовые поля были видны. Для этого нужно в панели **Properties** (Инспектор свойств) у текстового поля выставить опцию – **Show border around text** (Показывать рамку вокруг текста) (рисунок 7).

После этого создайте МС (нарисуйте прямоугольник или овал, выделите, нажмите F8, задайте **Movie Clip**). Задайте этому МС **Instance Name** (Имя экземпляра), например, **a** (рисунок 8).

Затем создайте кнопку, при действии на которую и будет происходить изменение свойств МС (нарисуйте прямоугольник или

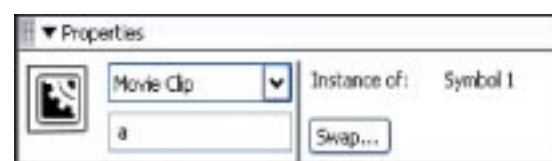


Рисунок 8.



Рисунок 7.

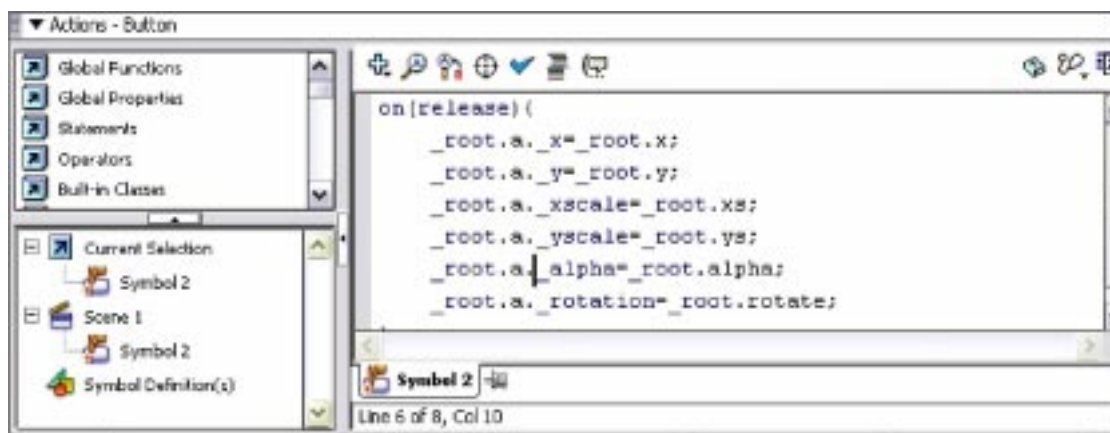


Рисунок 9.

овал, выделите, нажмите F8, задайте **Button**). Далее необходимо раскрыть панель **Actions** и написать следующий скрипт на кнопке:

```
on(release) {
    _root.a._x=_root.x;
    _root.a._y=_root.y;
    _root.a._xscale=_root.xs;
    _root.a._yscale=_root.ys;
    _root.a._alpha=_root.alpha;

    _root.a._rotation=_root.rotate;
}
```

В результате панель **Actions** будет выглядеть следующим образом (рисунок 9).

Отметим, что стоит всегда обращать внимание на заголовок панели **Actions**:



Если скрипт пишется Вами на кнопке и указывается обработчик событий кнопки – **on(release)**, то в заголовке должно быть указано слово **Button**, в противном случае возникнут ошибки, и скрипт работать не будет.

При помощи данного скрипта после введения числовых значений в текстовые поля и нажатия на кнопку, вы сможете менять свойства МС (рисунок 10).

Как вы понимаете, если возможно написать скрипт на кнопке, то возможно записать его и не каком-либо другом символе или на ключевом кадре. Самое простое, что можно придумать, это написание скрипта по изменению свойств

МС на самом МС (см. обработчики событий на МС из предыдущей статьи).

ЗАДАНИЕ ДВИЖЕНИЯ ОБЪЕКТУ БЕЗ ИСПОЛЬЗОВАНИЯ АНИМАЦИИ

Откройте новый файл и создайте МС, раскройте панель **Actions** и напишите в ней:

```
onClipEvent(enterFrame) {
    this._x+=5;
}
```

В результате действия данного скрипта, как только вы запустите ролик, созданный вами МС начнет двигаться самостоятельно без какой-либо анимации и уйдет за край видимой области. Секрет этого «чуда» вполне прост. Как вы помните, **onClipEvent** – это обработчик событий МС, **enterFrame** – одно из этих событий. Оно выполняется с частотой кадров вашего ролика (по умолчанию 12 кадров в секунду), то есть 12 раз в секунду скрипт будет вызывать некое действие, в данном случае будет выполняться изменение координат МС на 5 пикселей.

Недостаток предыдущего примера сводится к тому, что клип улетает за пределы экрана и пропадает навсегда. Необходимо доработать скрипт таким образом, чтобы ничего подобного не происходило. В данном случае можно пойти двумя путями. Первый – самый простой и бесхит-

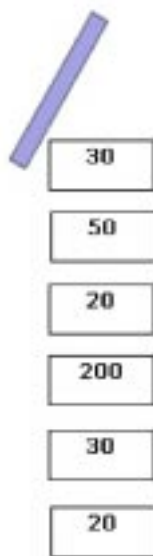


Рисунок 10.

ростный: необходимо устроить проверку, долетел ли МС до края сцены, и, если данное событие произошло, вернуть его, то есть МС, в нулевые координаты:

```
onClipEvent (enterFrame) {
    this._x+=5;
    if (this._x==550) {
        this._x=0;
    }
}
```

Вроде бы все хорошо, но необходимо вспомнить, какие значения были получены в примере с получением свойств МС. Как вы помните, МС может иметь координаты с ненулевой дробной частью, следовательно, условия равенства координат МС значению 550 будет выполняться крайне редко. Гораздо логичнее записать следующее:

```
onClipEvent (enterFrame) {
    this._x+=5;
    if (this._x>=550) {
        this._x=0;
    }
}
```

В этом случае условие будет выполняться всегда.

Второй пример гораздо интересней. Дело в том, что можно ввести следующий скрипт:

```
onClipEvent (load) {
    this.a=1;
}
onClipEvent (enterFrame) {
    this._x=5*this.a+this._x;
    if (this._x>=550) {
        this.a=-1;
    }
    if (this._x<=0) {
        this.a=1;
    }
}
```

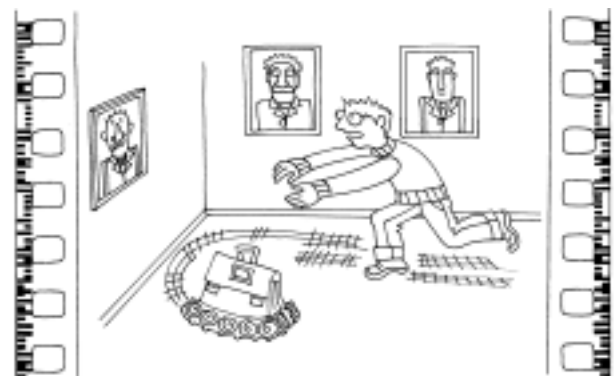
Итак, что же можно увидеть? Во-первых, появилось событие `onClipEvent (load)` – событие, записываемое на МС, которое происходит при загрузке данного МС, во-вторых, введена переменная `a`, эта переменная описана для данного МС, на это указывает слово `this`. Теперь ско-



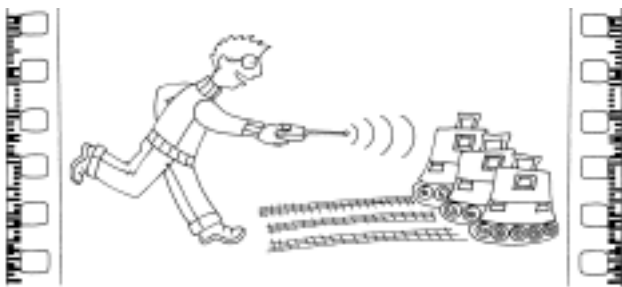
Недостаток ... примера сводится к тому, что клип учитывает за пределы экрана и пропадает навсегда.

пируйте МС и пять-семь раз вставьте в сцену. Посмотрите, что получилось: после запуска МС прямоугольники будут доходить до одной из границ клипа и затем будут поворачивать в противоположную сторону. Если ввести дополнительную переменную `b` и движение по координате `y` с пределами от 0 до 400, то получится следующий скрипт:

```
onClipEvent (load) {
    this.a=1;
    this.b=1;
}
onClipEvent (enterFrame) {
    this._x=5*this.a+this._x;
    this._y=5*this.b+this._y;
    if (this._x>=550) {
        this.a=-1;
    }
    if (this._x<=0) {
        this.a=1;
    }
}
```



...после запуска ... прямоугольники будут доходить до одной из границ клипа и затем ... поворачивать в противоположную сторону.



Теперь создайте кнопку, при нажатии на которую будет происходить дублирование ...

```

if(this._y>=400) {
    this.b=-1;
}
if(this._y<=0) {
    this.b=1;
}
}


```

Подобный скрипт, написанный на каждом из МС, будет приводить к тому, что при линейном движении МС по сцене он будет отталкиваться от мнимых стенок сцены и начинать движение в противоположном направлении.

`duplicateMovieClip();`

`duplicateMovieClip` – это создание копии конкретного МС с использованием **Instance Name** или указателя на текущий МС. Хотелось бы продемонстрировать несколько примеров создания и использования этого метода. Он присутствует в двух ипостасях: во-первых, осталась акция от скрипта Flash 4, которая находится в **Action**, во-вторых, это метод для управления МС.

Пример 1

Откройте новый файл, создайте МС, например, нарисовав квадратик при помощи инструмента **Rectangle** , выделите его при помощи инструмента **Selection Tool**

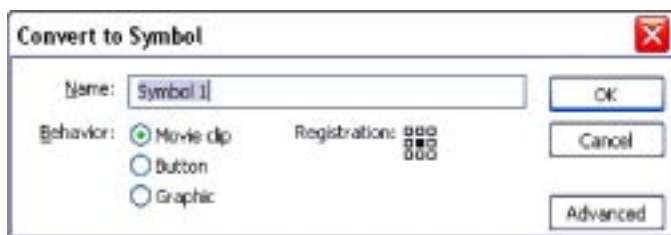



Рисунок 11.

 и нажмите горячую клавишу F8, выбрав при этом **Movie Clip** (рисунок 11).

Для дублирования МС необходимо, как и говорилось выше, задать **Instance Name**, что легко осуществить в панели **Properties**. Для примера назовем только что созданный МС "box" (рисунок 12).

Теперь создайте кнопку, при нажатии на которую будет происходить дублирование МС. Для этого нарисуйте квадрат, выберите инструмент **Selection Tool** и выделите получившийся квадрат, после чего необходимо нажать горячую клавишу F8 и выбрать пункт **Button** (Кнопка).

Для вызова функции `duplicateMovieClip` при нажатии на кнопку, скрипт должен иметь вид:

```

on (release) {
    duplicateMovieClip("box",
                        "box2", 1);
    box2._x=200;
}

```

"box" – **Instance Name** дублируемого МС должно полностью совпадать с именем, которое было задано в поле **Instance**.

"box2" – имя нового МС

1 – **Depth** – глубина нового МС, глубина с номером 0 занята оригиналом. Обратите внимание на то, что в приведенном случае после запуска тестового МС при нажатии на кнопку у вас не произойдет видимого изменения на сцене, поскольку один МС с именем "box2" будет продублирован в позицию с теми же координатами, что и оригинал, то есть необходимо для наблюдения изменить координату при помощи задания свойств `_x` и `_y`.

Пример 2

На кнопке необходимо дописать существующий скрипт, кото-

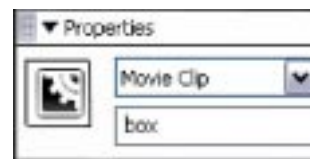


Рисунок 12.

рый должен теперь выглядеть следующим образом:

```
on (release) {
    duplicateMovieClip("box",
                       "box2", 1);
    _root.box2._x=200;
}
```

Вышеприведенное относилось к устаревшему способу дублирования МС, в настоящее время логичнее применять метод `duplicateMovieClip` к клипу "box", используя который, можно записать следующий скрипт:

```
on (release) {

    _root.box.duplicateMovieClip("box2",
                                  1);
    box2._x=200;
}
```

Запишите этот скрипт на кнопке. Таким образом, был продублирован один МС, теперь необходимо создать целую серию продублированных объектов.

Пример 3

Создайте скрипт, при помощи которого можно продублировать сразу несколько МС. При этом же необходимо посмотреть циклы:

```
on (release) {
    count = 1;
    while (count<=20) {
        _root.box.duplicateMovieClip("box"+count,
                                      count);
        count++;
    }
}
```

Как видно из приведенного скрипта, используется цикл `while`, который проверяет выполнение условия `count<20`, аналогичным образом можно применить цикл `for`, который автору, в силу привычки, нравится больше.

```
on (release) {
    for (count=1; count<=20;
        count++) {

        _root.box.duplicateMovieClip("box"+count,
                                      count);
    }
}
```

Выражение «box»+count выполняет функцию динамического задания имени для создания новых МС, count – выполняет роль счетчика, увеличивающегося от 1 до 20, и в то же время задает глубину на которую помещается новый МС. Для каждого МС должна быть своя глубина – `depth`. Хочется отметить, что для динамического получения имени продублированного МС можно применять также следующие варианты

`Newname="box"+String(count);`

или

`Newname="box"&counter;`

Пример 4

Этот пример демонстрирует использование случайных значений (рандомизации) с использованием функции `random()` и задания при помощи введенной рандомизации случайных свойств продублированных МС. Каждый продублированный МС получает свои случайные свойства:

`_x` – позиция по оси X,

`_y` – позиция по оси Y,

`_xscale` – масштабирование по оси X,

`_yscale` – масштабирование по оси Y,

`_alpha` – прозрачность

Приведенный ниже скрипт демонстрирует управление указанными свойствами МС. Его необходимо написать на кнопке, при нажатии на которую у вас происходит дублирование МС.

```
on (release) {
    count = 1;
    while (count<20) {
        _root.box.duplicateMovieClip("box"+count,
                                      count);
        _root["box"+count]._x =
            random(550);
        _root["box"+count]._y =
            random(400);
        _root["box"+count]._xscale =
            random(150);
        _root["box"+count]._yscale =
            random(150);
        _root["box"+count]._alpha =
            random(100);
        count += 1;
    }
}
```



В данном примере использовано динамическое задание имени...

В данном примере использовано динамическое задание имени с использованием адресации – `_root[<box>+count]`. Если раньше вы привыкли писать имена MC после точки, то для динамического использования имени необходимо имя писать в квадратных скобках. Подобные записи используются при задании массивов (ассоциативных массивов). Если задать анимацию для вашего MC, то вы сможете наблюдать очень симпатичную картинку на экране (либо войти в режим редактирования символа, либо задав скрипт, как описано в одном из приведенных выше примеров).

attachMovieClip

attachMovie – это метод, загружающий и запускающий MC из **Library** (Библиотеки), как только поступил запрос. При этом MC, загруженный на сцену, получает свое уникальное имя, по которому к «приаттаченному» MC можно будет обратиться и заставить его выполнить определенную последовательность действий или задать то или иное свойство и глубину (**depth**). В общем случае обращение к MC выглядит следующим образом:

`MCInstanceName.attachMovie(idName, newname, depth);` где:



Рисунок 13.

MCInstanceName – это **Instance name** MC, внутри которого «приаттачивается» MC, **idName** – это имя «приаттачиваемого» MC в библиотеке (**Library**). Если библиотека недоступна, то необходимо ее открыть последовательностью действий: **Window** → **Library**. Это имя возможно установить двумя способами: во-первых, имя можно задать при создании символа. Выберите инструмент **Rectangle** и нарисуйте квадрат, затем **Selection Tool** выделите квадрат, нажмите F8 и увидите хорошо знакомое диалоговое окно (рисунок 13).

В правом нижнем углу вы увидите кнопку **Advanced**, необходимо нажать на нее (рисунок 14).

В открывшемся дополнительном поле необходимо обратить внимание на слово **Linkage**. Поставив галочку в пункте **Export for ActionScript**, сделаем MC доступным для экспортирования из библиотеки (рисунок 15).

Как только вы поставите галочку в пункте **Export for ActionScript**, автоматически ставится галочка в пункте **Export in first frame** и становятся доступными для редактирования два текстовых поля **Identifier** (поле, в которое необходимо ввести **idName** для последующего использования) и **AS 2.0 Class** (данное поле будет рассмотрено позднее в рамках раздела объектно-ориентированного программирования).

Если Вы забыли указать **idName** при создании MC, то это можно сделать из библиотеки, щелкнув ПКМ на требуемом MC из контекстного меню параметр **Linkage**, после чего увидите уже описанное выше



Если библиотека недоступна, то необходимо ее открыть последовательностью действий: Window → Library.

диалоговое окно и сможете задать **idName** (рисунок 16).

depth – целочисленная спецификация глубины нахождения МС (**depth**).

Создайте новый файл и создайте три кнопки и три МС, которым задайте **idName** "att1", "att2", "att3" по описанной выше методике. Удалите их со сцены **Linkage** – "att1", "att2", "att3". На кнопках напишем:

```

Кнопка 1
on (release) {
    _root.attachMovie("att1",
        "newname1", 1);
}

Кнопка 2
on (release) {
    _root.attachMovie("att2",
        "newname2", 1);
}

Кнопка 3
on (release) {
    _root.attachMovie("att3",
        "newname3", 1);
}
    
```

По каждой кнопке указывается глубина 1 **depth(1)**, вследствие чего и происходит загрузка и отгрузка МС. Используйте **removeMovieClip** или **unloadMovie** акции и методы для принудительной отгрузки МС, например, написав на еще одной кнопке:

```

on (release) {
    _root.newname1.unloadMovie();
}
    
```

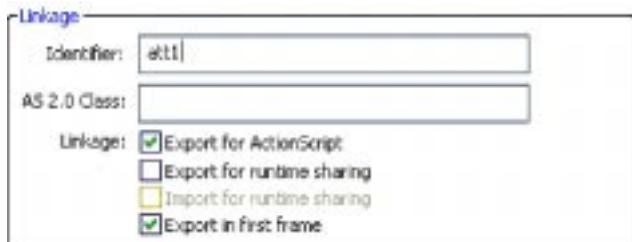


Рисунок 15.

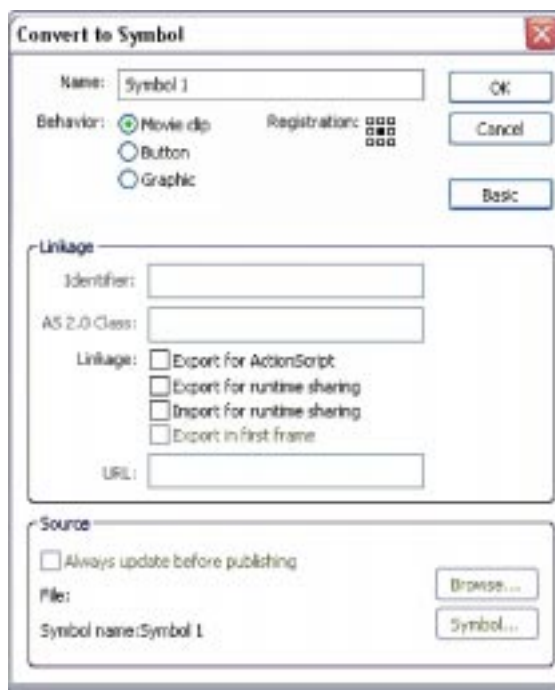


Рисунок 14.

Задание для самостоятельной работы. Создайте несколько МС со случайными свойствами и повторите созданное при помощи **duplicateMovieClip**.

Вот и закончилось знакомство с основными свойствами МС и еще с двумя интересными методами, позволяющими держать в библиотеке минимальное количество символов и тем самым эффективно использовать возможности Flash.

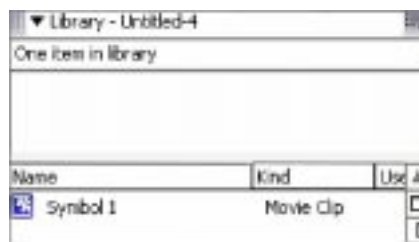


Рисунок 16.

*Штенников Дмитрий Геннадьевич,
доцент кафедры физики
СПбГУ ИТМО.*



Наши авторы, 2004.
Our authors, 2004.