

MACROMEDIA FLASH. ОСНОВЫ ПРОГРАММИРОВАНИЯ. ПРОСТЕЙШИЕ СКРИПТЫ. УРОК 2

ОСНОВНЫЕ СОБЫТИЯ MOVIE CLIPS

Большая доля работы во Flash приходится на манипуляцию символами. Практически все базовые приемы, все трюки и эффекты невыполнимы без этих действий. С помощью сценариев на ActionScript вы можете выполнять практически любые действия над символами. Надо только помнить, что выполнить эти действия можно либо в ответ на действие пользователя, либо при наступлении какого-то кадра на временной шкале.

Создание Movie Clip'а ничем не сложнее кнопки. Необходимо выполнить все действия по созданию кнопки, за исключением того, что в диалоговом окне по выбору типа символа необходимо выбрать **Movie Clip** (мультик).

Movie Clip (MC) могут реагировать на следующие действия – **onClipEvent()**:

load – действие происходит как только загружается MC – одно из самых частых используемых действий;

unload – действие происходит как только выгружается MC;

enterFrame – инструкция выполняется со скоростью вашего фильма, сколько вы поставили кадров в секунду – с такой скоростью все и будет происходить;

mouseMove – инструкция выполняется по движению мышки;

mouseDown – инструкция выполняется по нажатию левой клавиши мышки;

mouseUp – инструкция выполняется по отпусканю левой клавиши мышки;

keyDown – инструкция выполняется по нажатию клавиши на клавиатуре;

keyUp – инструкция выполняется по отпусканю клавиши на клавиатуре;

data – инструкция выполняется по получению данных в MC.

События выполняются при помощи директивы **onClipEvent()**. Синтаксис ее таков:

```
onClipEvent (событие)
{
  ... // Инструкции
}
```

Например, остановка MC:

```
onClipEvent (load)
{
  stop();
}
```

ОСНОВНЫЕ ДЕЙСТВИЯ С MOVIE CLIP

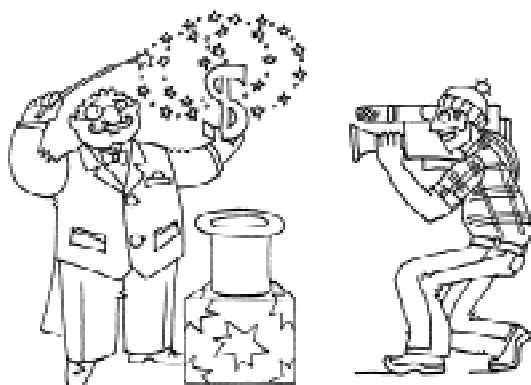
Основными действиями с MC, как правило, являются:

play() – начинает или возобновляет воспроизведение клипа;

stop() – останавливает воспроизведение клипа;

gotoAndPlay() – переходит на определенный кадр (сцену) и продолжает воспроизведение;

gotoAndStop() – переходит на определенный кадр (сцену) и останавливает воспроизведение.



Большая доля работы во Flash приходится на манипуляцию символами.

Свойства (параметры) клипов, которые можно считывать или изменять:

_x, **_y** – координаты клипа (в пикселях);

_xscale, **_yscale** – масштаб клипа (в процентах), соответственно, по горизонтали и по вертикали;

_width, **_height** – ширина и высота клипа (в пикселях);

_rotation – угол поворота клипа (в градусах);

_alpha – непрозрачность клипа (в процентах) – измеряется от 0 до 100; если значения не укладываются в указанные пределы, то они округляются либо до 100, либо до 0;

_visible – видимость.

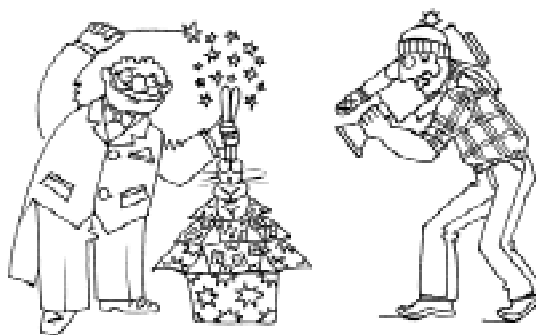
Это далеко не все, что можно делать с клипами. Используйте другие параметры.

ИМЕНА

Для того чтобы обращаться к клипам, вам потребуется разобраться с понятием имени объекта (**Instance name**) и пути до объекта (**Target path**). Договоримся, что клип (*movie clip*) и объект для нас – одинаковые вещи.

Имя объекта – это имя конкретного экземпляра символа. Скажем, у нас может быть символ – машинка, а экземпляры этого символа будут называться «Машинка1», «Машинка2», «Pickup», «Запорожец»...

В принципе существует возможность управлять символами и без использования имен объектов. Всем символам имя назначается автоматически при создании: *instance1*, *instance2*, ... *instancen*. По этой причине вы можете и не называть MC как-то особенно, но для лучшей управляемости Flash ролика целиком задание **Instance name** особенно рекомендуется (рисунок 1).



...объекты можно «вкладывать» друг в друга...

Для того чтобы дать имя объекту, нужно выделить объект и в панели **Instance (Window → Properties)** в поле **Instance Name** ввести имя объекта. Имена могут состоять только из букв, цифр и символа подчеркивания («_»), причем имя не может начинаться с цифры.

ПУТИ

Путь до объекта – это запись имени объекта с учетом иерархии. Вы знаете, что Flash-объекты можно «вкладывать» друг в друга, создавая, таким образом, иерархию. Эта вложенность обеспечивает не только удобство в обращении с объектами, но и ограничивает видимость имен объектов. Видимость ограничивается своим уровнем. Объект может напрямую (по имени) обращаться только к объектам, входящим в него, стоящим на 1 уровень ниже в иерархии.

Для того чтобы обратиться к объекту другого уровня, нужно знать путь до него. Причем путь может указываться как абсолютно (с самого верхнего уровня иерархии), так и относительно (с текущего уровня).

Путь включает в себя объекты, через которые нужно «пройти» по дереву иерархии, чтобы добраться до нужного нам объек-



Рисунок 1.

та. Имена объектов перечисляются через точку. Кроме того, существует несколько указателей (можно их назвать «виртуальными объектами»), которые часто очень полезны:

this – указатель на самого себя (то есть на текущий объект). Нужен, например, для передачи в функцию указателя на объект, из которого эта функция вызывается, или для обработки переменной или свойства внутри текущего объекта.

_parent – указатель на «родителя». Указывает на объект, стоящий в иерархии одним уровнем выше.

_root – «корень». Это начало иерархии. Без него не обойтись при указании абсолютного пути.

_level0, _level1, ..., leveln – указатель на начало иерархии ролика, загруженного на уровень 0,1, ..., n.

_global – аналогично **_root**, рассмотрение всех его тонкостей в рамке данной статьи не укладывается.

Путь выглядит так:

leaf.play(); – у подобъекта **leaf** (лист) вызывается функция **play()**;

_parent.tree.leaf.stop(); – подразумевается, что на одном уровне имеется объект **tree**, у которого есть объект **leaf**, у которого и вызывается функция **stop()**;

_root.banner._visible = false; – сделать клип **banner**, находящийся на 1-ом уровне, невидимым.

ОТЛАДКА В ACTIONSCRIPT

Окна **Output** (вывод) и **Debugger** (отладчик). Это инструменты, служащие для отладки сценариев ActionScript.

Окошко **Output** пришло из Flash 4, где оно было единственным инструментом для отладки. Существует директива **trace()**, которая выводит сообщения в это окошко. Туда же выводятся сообщения об ошибках.

Использовать **trace()** очень просто:

trace («280-й кадр»);

или, например,

trace (xpos + k);

С 5-ой версии Flash появился специальный инструмент – окошечко **Debugger**.

Чтобы им пользоваться, нужно проверять свои фильмы не как обычно – **Test movie** (Ctrl + Enter), а с помощью **Debug movie** (Ctrl + Shift + Enter). Окошко **Debugger** можно скрыть/показать с помощью **Window → Debugger**.

КАЧЕСТВО

Вы, наверное, уже знакомы с понятием качества во Flash (опция **Quality** при публикации, параметры **_quality, _highquality**, функция **toggleHighQuality()**). Качество тоже сильно влияет на скорость воспроизведения. Тут также существует несколько компромиссов.

Во-первых, уровень качества можно установить вручную при экспорте. При этом не забывайте, что если не отключить контекстное меню, то у пользователя остается возможность регулировать качество.

Во-вторых, качество можно менять динамически во время исполнения анимации (параметр **_quality**). На время воспроизведения особенно сложных и «быстрых» фрагментов можно понижать качество, тем самым выигрывая в скорости, а когда «количество» анимации уменьшится, вновь возвратиться к высокому уровню качества.

При создании анимационных заставок можно давать выбор качества пользователю, поместив в уголок пару-тройку кнопок, позволяющих регулировать качество.

Поэкспериментируйте, и вы увидите, что качество существенно влияет на скорость воспроизведения клипов.

СОЗДАНИЕ МЕНЮ ПРИ ПОМОЩИ MC.PLAY();

Выберите инструмент «квадратик» и нарисуйте прямоугольник, после этого выберите инструмент «черн. стрелочка» и выделите прямоугольник, при этом прямоугольник выделяется «сеточкой». После этого выберите пункт меню **Insert → Convert to Symbol** (или нажмите горячую клавишу F8). В появившемся диалоговом окне выберите тип символа (**Behavior**) **Button**. Полученная кнопка выделяется голубой рамкой.

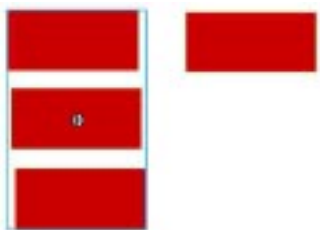


Рисунок 2.

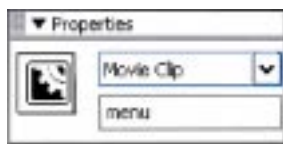


Рисунок 3.




Рисунок 4.

Скопируйте эту кнопку в буфер обмена (**Edit → Copy** или **Ctrl + C**). Два или три экземпляра этой кнопки вставьте на сцену (**Edit → Paste** или **Ctrl + V**) и расположите по сцене, выровняв их друг относительно друга. Эти две-три кнопки и будут выполнять роль подпунктов меню. При помощи курсора мыши выделите вставленные кнопки (рисунок 2).

Преобразуйте выделенные кнопки в новый символ – **MovieClip** (**Insert → Convert to Symbol** или **F8**). Три кнопки объединятся в одну рамочку и будут представлять собой единый символ, обладающий набором свойств как объект в программировании. В дальнейшем для обращения к этому movie-клипу (МС) желательно задать ему имя (**Instance name**). При этом имя МС не должно начинаться с цифр. Для примера назовем МС «menu» (рисунок 3).

Нажмите правой клавишей мыши на одну из кнопочек, входящих в состав МС, и выберите пункт контекстного меню **Edit in place** (или щелкните 2 раза левой кнопкой

мыши (ЛКМ)), и вы перейдете в режим редактирования МС. Как вы помните, каждая отдельная анимация осуществляется в отдельном слое. Поэтому необходимо создать число слоев на один меньше, чем количество кнопок в МС, для того чтобы задать корректно анимацию для каждой из них. Если кнопка у нас три, то соответственно требуется создать еще два слоя (рисунок 4).

Чтобы создать необходимое число слоев, нужно в панели времени найти кнопку  и щелкнуть по ней ЛКМ два раза. При этом временная линейка примет вид, как на рисунке 5.

Видно, что все три кнопки находятся в первом слое, а слои 2 и 3 пустые. Переместим каждую кнопку в свой слой. Для этого выделим одну из трех кнопок и вырежем ее в буфер обмена (**Edit → Cut** или **Ctrl + X**). Нажатием ЛКМ перейдем в первый ключевой кадр второго слоя и вставим кнопку на то же место на сцене, откуда мы ее скопировали (**Edit → Paste in Place** или **Ctrl + Shift + V**). Вторую кнопку аналогичным образом переместим в третий слой. Таким образом, **Timeline** примет вид, как на рисунке 6.

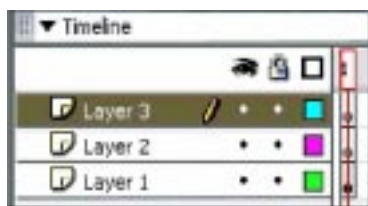


Рисунок 5.

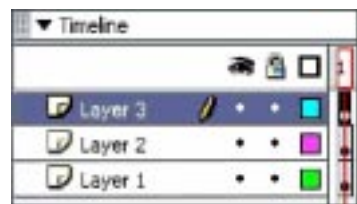


Рисунок 6.



...выделите прямоугольник, при этом прямоугольник выделяется «сеточкой».

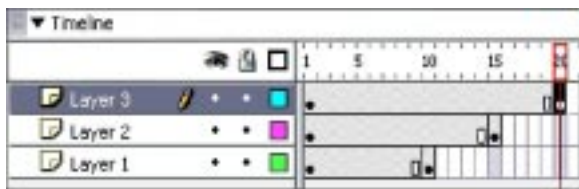


Рисунок 7.

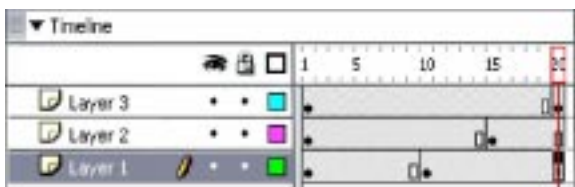
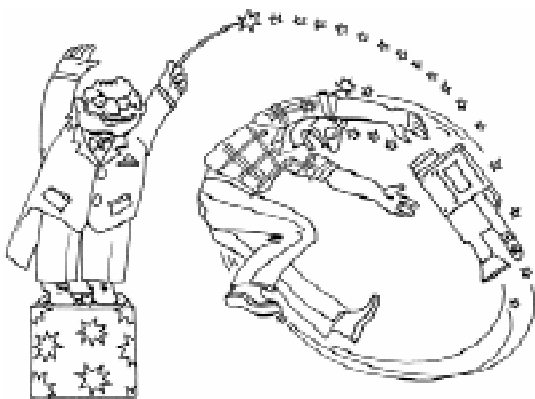


Рисунок 8.

Создадим ключевые кадры в следующем порядке: в первом слое – на 10 кадре (нажмите ЛКМ на 10 кадре первого слоя и выберите пункт меню **Insert → Keyframe**, или нажмите горячую клавишу F6, или нажмите ПКМ и в контекстном меню выберите **Insert Keyframe**), во втором – на 15 и в третьем – на 20 кадре (рисунок 7).

При этом вы будете видеть всего одну кнопку в МС. Это связано с тем, что время существования первого и второго слоев меньше, чем у третьего слоя. Следовательно,



...выберем анимацию ... и поставим одно вращение по часовой стрелке...



Рисунок 9.

но, продлим время существования первого и второго слоев до 20 кадра (нажмите ЛКМ на 20 кадре и выберите пункт меню **Insert → Frame**, или нажмите горячую клавишу F5, или нажмите ПКМ и в контекстном меню выберите **Insert → Frame**) – на экране появятся все три кнопки. А **Timeline** примет вид, как на рисунке 8.

Нажатием ЛКМ по первому ключевому кадру в первом слое перейдите в него, при этом у вас выделится первая (самая верхняя) кнопка, щелкните по ней повторно ЛКМ.

Откройте панель **Properties** (инспектор свойств), если ее нет на экране, выполните эту операцию следующим образом: **Window → Properties** (рисунок 9).

Найдите раздел **Color** и выберете среди представленных вариантов пункт **Alpha**: 100% – объект непрозрачен, 0% – объект прозрачен полностью. Выставьте полную прозрачность объекта, при этом, вместо кнопки, вы увидите прозрачную рамку. Аналогичные операции повторите для второй и третьей кнопок. При этом все три кнопки у вас исчезнут, и в первом кадре вы будете видеть крестик, символизирующий собой центр редактируемого МС.

Зададим анимацию. Для этого нажмем ЛКМ на первом кадре в первом слое, и в панели **Properties** выберем анимацию **Motion (Tweening → Motion)**, и поставим одно вращение по часовой стрелке (**Rotate → CW, 1**) (рисунок 10).

Во втором слое все то же самое, но два вращения, в третьем слое – три вращения.

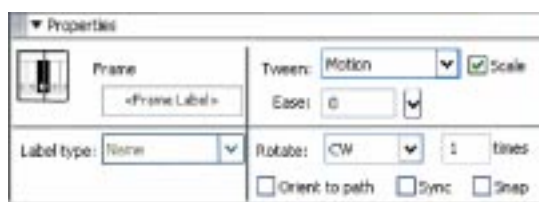


Рисунок 10.



Рисунок 11.



Рисунок 12.

Как вы помните, Flash имеет особенность, состоящую в том, что проигрывание анимации происходит автоматически сразу после загрузки ролика и продолжается до бесконечности. Следовательно, возникает необходимость принудительно заставить открываться меню по вашему желанию. Это можно осуществить следующим образом. Нажмите ЛКМ на 1 ключевом кадре третьего слоя и откройте панель **Actions**. Нажмите на знак «+», выберите **Global Functions** → **Timeline Control** → **stop**, при этом в поле для скрипта у вас появится следующая запись:

```
stop();
```

Аналогичную операцию повторите для 2 ключевого кадра в третьем слое.

С меньшими усилиями эту команду можно было записать просто руками.

Вернемся на сцену 1 (нажатию на надпись **Scene 1** в верхнем левом углу над Timeline). При этом вы увидите, что три пункта подменю не видны, видна только рамка голубого цвета с кружком посередине (рисунок 11).


Теперь необходимо на кнопку, являющуюся пунктом меню, написать скрипт,

управляющий МС. Для этого нажмите ЛКМ на кнопке и раскройте панель **Actions**. Сделайте следующую надпись:

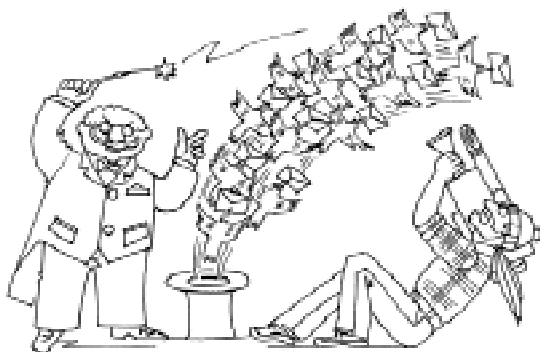
on (после чего у вас появится список возможных вариантов, на что именно может откликнуться ваша кнопка (рисунок 12)).

Как правило, выбирают **release**, то есть по отпусканию. Продолжайте писать дальше:

```
on (release) {
    .play();
}
```

Необходимо указать имя объекта, к которому будет применен метод **play()**. Поставьте курсор перед точкой и в панели **Actions** найдите хорошо известную вам кнопку , при этом появится следующее диалоговое окно (рисунок 13).

Для добавления объекта, к которому будет применен метод **play()**, нажмите ЛКМ на слове «menu». Обратите внимание, что здесь возможны два варианта обращения к МС: абсолютная адресация (**Absolute**) и относительная (**Relative**). В нашем случае выберем абсолютную адресацию и нажмем ОК. В дальнейшем при освоении про-



...выберет абсолютную адресацию и нажмет ОК.



Рисунок 13.

граммирования желательно от абсолютной адресации отойти, но это будет потом.

```
on (release) {
  _root.menu.play();
}
```

Все. Теперь просмотрите готовое меню при помощи Ctrl + Enter.

СОЗДАНИЕ МЕНЮ ЧЕРЕЗ MC.NEXTFRAME();

Выполните все операции из предыдущего примера до момента введения **Instance Name**, для данного примера назовем MC «menu2».

Нажмите ПКМ на одной из кнопок, входящих в состав MC, выберите пункт контекстного меню **Edit in place**, и вы перейдете в режим редактирования MC.

Создайте 2 ключевой кадр нажатием клавиши F6. У вас получатся два рядом стоящих ключевых кадра (рисунок 14).

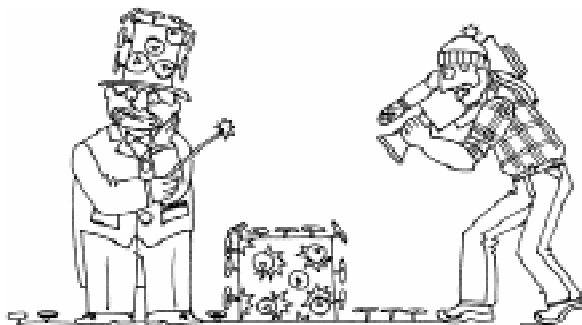
Нажатием ЛКМ в 1 ключевой кадр MC перейдите на него и нажмите клавишу Delete, удалив содержимое 1 ключевого кадра (подпункты меню) (рисунок 15).

Нажмите ЛКМ в 1 ключевом кадре и выберите **Actions**. Далее в поле для ввода скрипта напишите следующую запись:

```
stop();
```


Вернитесь в режим редактирования сцены. Вместо MC, вы увидите кружок.

Напишем управляющий скрипт на кнопке. Наждем ЛКМ на единственно видимой кнопке (если кнопок больше, чем одна, значит вы сделали что-то неправильно) и выберем пункт **Actions**. После чего напишем:



...если кнопок больше, чем одна, значит вы сделали что-то неправильно...

```
on (release) {
  .nextFrame();
}
```

nextFrame – это метод, осуществляющий переход на следующий кадр. Теперь необходимо указать имя объекта. Для этого аккуратно нажмите ЛКМ перед точкой. Нажмите кнопку . По нажатию на нее появится список объектов, из которых мы выбираем «menu2». Выберем абсолютную адресацию и нажимаем **OK**.

```
on (release) {
  _root.menu2.nextFrame();
}
```

Проверим работоспособность этого скрипта (Ctrl + Enter). По нажатию на пункт меню появится подменю. По повторному нажатию на него же ничего не произойдет.

Требуется доработать скрипт так, чтобы при повторном нажатии подпункты меню исчезали. Для этого перейдите в режим редактирования скрипта на управляющей кнопке. Строчкой ниже открытых фигурных скобок **if** и в условии для **if (condition)** напишите **!a.** и после этого откройте фигурные скобки до **_root.menu.nextFrame()**, а затем закройте таким образом, чтобы скобка находилась внутри фигурных скобок после **if**.

Затем напишите **else** и снова откройте фигурные скобки. После этого, напишите **_root.menu2.prevFrame()**; и закройте фигурные скобки.

if(!a) – это оператор условия, в скобках указывается само условие. В данном случае проверяется **!a.**

prevFrame() – это метод, осуществляющий переход на предыдущий кадр;



Рисунок 14.



Рисунок 15.

! означает логическое отрицание;
a – переменная. Интересно отметить, что в ActionScript логические переменные могут принимать следующие значения:

true – истина;
false – ложь.

Но, помимо перечисленных выше значений, логические переменные во Flash могут принимать значения 1 и 0, соответственно. Этой особенностью мы и воспользуемся.

Теперь необходимо изменить значение переменной **a**, которую мы обрабатываем в условии **if**. Для этого перейдем к месту перед последней закрывающей фигурной скобкой и напишем:

```
a=1-a;
```

Также хочется отметить, что одна и та же переменная может выступать в роли логической, целочисленной и строковой переменной одновременно без какого-либо переопределения.

В результате проведенной доработки скрипта было получено следующее:

```
on (release) {  

  if (!a) {  

    _root.menu2.nextFrame ();  

  } else {  

    _root.menu2.prevFrame ();  

  }  

a=1-a;  

}
```

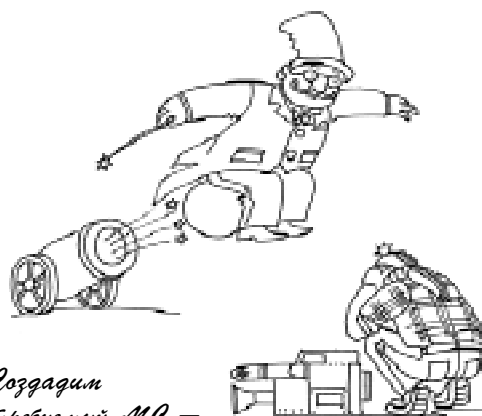
Дело за малым – задать начальное значение переменной **a** (в предыдущих версиях Flash такой операции не требовалось). Для этого необходимо щелкнуть ЛКМ по первому ключевому кадру сцены и в панели **Actions** написать:

```
a=0;
```

У вас получилось работающее меню.

Доработаем получившееся меню так, чтобы по нажатию на подпункт меню запускался созданный вами МС.

Создадим требуемый МС – например, движущийся шарик. Для этого выберем инструмент «кружок» и нарисуем на сцене шарик. Преобразуем его в МС – **Insert** → **Convert to Symbol** (или F8) и дадим ему имя «**klip**». Перейдем внутрь МС (**контекстное меню** → **Edit In Place**) и зададим для кружка **Shape**-анимацию.




Создадим требуемый МС – например, движущийся шарик.

Перейдите внутрь МС «**menu2**» на второй ключевой кадр (в котором содержатся подпункты меню). Выберите одну из кнопок и вызовите панель **Actions** (**контекстное меню** → **Actions**). В окне скрипта необходимо написать:

```
on (rollover) {  

  .play ();  

}
```

Поставьте курсор перед **.play ()**; и выберите путь к МС «**klip**» при помощи кнопки  в верхней части панели **Actions**.

```
on (rollover) {  

  _root.klip.play ();  


}
```

Запустите ваш МС. Теперь у вас при нажатии на подпункт меню выполняется МС с именем «**klip**», но при этом подпункты меню остаются на экране.

ТЕКСТОВЫЕ ПОЛЯ

Для дальнейшей работы желательно получить некоторые сведения о текстовых полях. В текстовые поля может производиться ввод и вывод значений переменных. Возможны два варианта ввода и вывода.

Самый первый и самый традиционный является заданием переменной. Рассмотрим, как это делается.

Выберите инструмент **Text Tool**  и посмотрите на Инспектор свойств (панель **Properties**) (рисунок 16).


Правее буквы «**A**» находится вариант выбора текстовых полей. Здесь возможны три

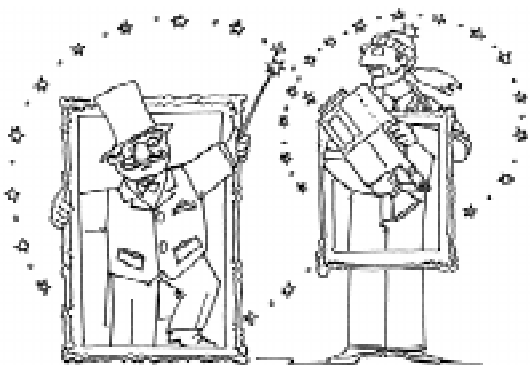


Рисунок 16.

варианта: **Static Text** – служит для ввода текста на сцену, как в любом текстовом графическом редакторе, **Dynamic Text** – динамическое текстовое поле, данный режим позволяет только выводить данные, и **Input Text** – позволяет вводить и выводить данные. Хочется обратить ваше внимание на кнопку  – **Show border around text**, которая позволяет создавать рамки вокруг полей.

Для примера необходимо выставить два текстовых поля на сцену (рисунок 17).

В свойствах обоих текстовых полей можно увидеть поле **Var**, в нем необходимо написать имя переменной, которая будет вводиться или выводиться в это поле. Для этого необходимо выбрать инструмент **Selection Tool**  и щелкнуть ЛКМ по текстовому полю, после чего в поле **Var** написать имя переменной, например «a».



...Show border around text, которая позволяет создавать рамки вокруг полей.



Наши авторы, 2004.
Our authors, 2004.

Для второго текстового поля необходимо ввести то же название переменной. В результате после запуска ролика текст, введенный в верхнее текстовое поле, будет появляться в нижнем, и наоборот.

Следующий пример чуть более сложный. Повторите описанные выше действия до момента назначения имен переменных. И для первого текстового поля введите переменную «a», а для второго «b». После этого создайте кнопку, а в скрипте на кнопку необходимо будет записать следующее:

```
on(release) {
  _root.a=_root.b;
}
```

В данном примере, поскольку оба текстовых поля находятся на сцене, можно написать лишь:

```
on(release) {
  a=b;
}
```

В принципе возможна и следующая запись на кнопке:

```
on(release) {
  a.text=b.text;
}
```

Но такая запись возможна лишь в том случае, когда «a» и «b» – не имена переменных, а **Instance Name** (Имя объекта) текстового поля. Это как раз второй способ ввода и вывода, о котором речь шла выше.

Как видите, значение из текстового поля с переменной «b» передается в текстовое поля с переменной «a».

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