Штенников Дмитрий Геннадьевич

MACROMEDIA FLASH. ОСНОВЫ ПРОГРАММИРОВАНИЯ. ПРОСТЕЙШИЕ СКРИПТЫ. УРОК 5

В предыдущей статье началось описание и обсуждение простых приложений, созданных с использованием языка программирования ActionScript. Вданной статье мы рассмотрим уроки по созданию приложений с использованием событий.

ПРИМЕРЫ НА ИСПОЛЬЗОВАНИЕ СОБЫТИЙ

Для начала вспомним о том, что такое событие. Большинство компьютерных программ нацелены на обработку того или иного события, инициированного пользователем, например захват и отпускание того или иного объекта, изменение размеров окна. Все возможности интерактивности определяются теми путями, которые были заложены в программном обеспечении во время его создания. Эти возможности являются следствием различных событий (нажатие на клавишу мышки, движение курсора мышки по экрану, нажатие клавиш на клавиатуре и т. д.)

События позволяют организовать интерактивную составляющую вашего ролика контроливанием при помощи скриптов. Каждый скрипт вашего ролика реагирует на событие, как и в большинстве программ. Вслучае Movie Clip (MC) такими событиями, например, могут являться наведение курсора мышки на кнопку или нажатие клавиши, а также переход MC в тот или иной кадр и т. д.

В ActionScript описания (обработчик) событий (за исключением событий, совершаемых по переходу в ключевой кадр) обычно записываются первой же строчкой скрипта, например:

```
When this happens (event) {
  do this;
  do this;
}
```

С событиями, записанными в ключевых кадрах, все несколько иначе, поскольку, если поместить скрипт в ключевой кадр, нет необходимости дополнительно идентифицировать событие кадра, так как скрипт начинает свое выполнение, как только ваш MC перейдет на кадр со скриптом, и, следовательно, нет необходимости дополнительно это событие описывать. Скрипт, записанный в ключевом кадре, примерно выглядит следующим образом:

```
do this;
do this;
```

Для того чтобы понять, что такое событие, нет ничего лучше, чем опыт по созданию приложений. К этому мы и приступаем.

ВЫБОР ОПТИМАЛЬНОГО СОБЫТИЯ И ОБРАБОТЧИКА СОБЫТИЯ

Использование обработчиков событий – наиболее серьезный выбор программиста ActionScript. Возможно, это является следствием того, что достаточно сложно определить, какое из событий, описанных стандартно, необходимо применить в том или ином случае, а в отсутствие оных,

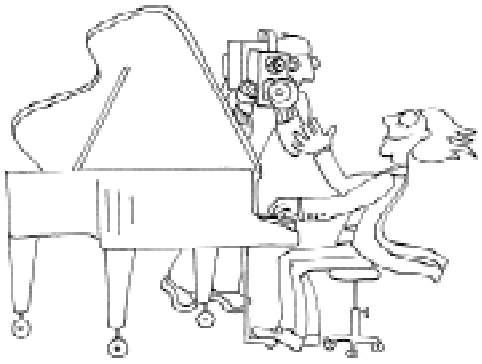
Рисунок 1.

какое событие требуется дополнительно описать.

В данных примерах будет представлен ряд путей, которые можно применить в том или ином случае.

ИСПОЛЬЗОВАНИЕ СОБЫТИЙ МЫШИ

Событие нажатия на левую клавишу мыши (ЛКМ) (правая клавиша мыши зарезервирована для встроенного меню, и



избавиться от этого очень сложно) взаимодействуют с кнопками и МС – позже мы опишем исключения из данного описания.

Если вы ранее работали с Flash 4 и 5, то помните, что на ЛКМ реагировали только кнопки, однако, начиная с Flash MX, на ЛКМ стали реагировать еще и МС. Рассмотрим стандартные события.

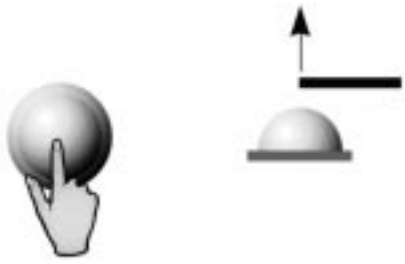
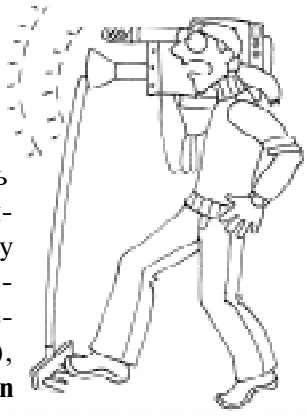


Рисунок 2.

Контакт (нажатие ЛКМ): on (press)



В физическом смысле – когда вы касаетесь чего-нибудь, например, кусочка льда, у вас происходит реакция: вы реагируете (вам холодно), кусочек льда тает; **on (press)** обработчик событий работает наилучшим образом для эмулирования касания, захвата, надавливания, как и комбинации этих действий. Вы можете использовать это событие для запуска скрипта, когда курсор находится над кнопкой или МС, и когда кнопка нажата (рисунок 1).

Отпускание: on (release)

Этот обработчик события подражает отпусканию объекта (например, отпускания после перетаскивания). Поскольку, если пользоваться **Normal mode** панели **Actions**, этот обработчик события вставляется по умолчанию – это самый легкий способ позволить пользователю заставлять ваш МС действовать. Вы можете использовать этот обработчик, чтобы вызвать скрипт в то время, когда кнопка мыши нажата и курсор мышки находится над кнопкой (рисунок 2).

Нажатие кнопки с перемещением: on (releaseOutside)

Данный обработчик события используется обычно, чтобы вызвать скрипт, когда пользователь нажал кнопку МС и переместил мышь, чтобы подражать перемещению или моментальному снимку. Также данный обработчик может использоваться для имитации раздачи карт во время карточной игры.

Управление с клавиатуры: on (keyPress)

Вы можете использовать этот обработчик, чтобы вызвать скрипт, когда пользователь нажимает символ, номер, знак

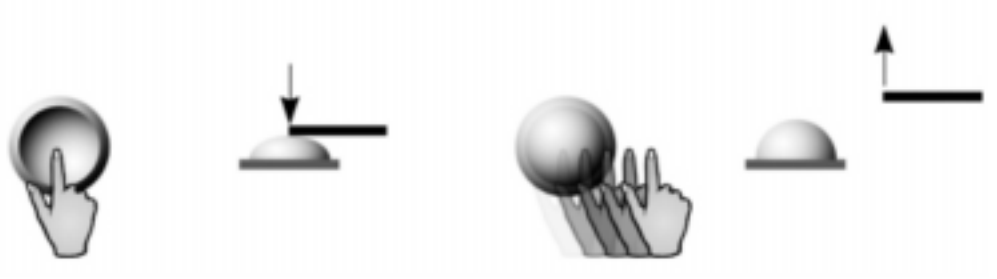


Рисунок 3.

препинания, стрелку или возврат на один символ, **Insert**, **Home**, **End**, **Page Up**, или **Page Down** (рисунок 3).

Наведение курсора мыши на кнопку: on (rollOver)

Если разместить вашу руку поверх горячей печи и чувствовать теплоту фактически без прикосновения к печи, то это будет аналогом данного обработчика. Данный обработчик события используется, чтобы подражать способу, которым объекты затрагивают другие объекты, которые излучают теплоту, холод, свет, являются кнопками или МС. Вы можете также использовать этот обработчик события, чтобы отобразить информацию о том, что кнопка или МС сделают, прежде чем они будут нажаты. Вы можете использовать этот обработчик случая, чтобы вызвать скрипт, когда пользователь размещает мышь на кнопке или МС.

Отведение курсора мыши: on (rollOut)

Очевидно, когда вы отодвигаете вашу руку от той печи, о которой говорилось в предыдущем абзаце, вы прекращаете чувствовать теплоту, которую излучает печь, и ваша рука остывает. Это то явление, которому этот обработчик случая подражает. Вы можете использовать этот обработчик случая, чтобы вызвать скрипт, когда пользователь перемещается с кнопки или МС (рисунок 4).

Перетаскивание: on (dragOver)

Действие данного обработчика событий влечет за собой передвижение объекта внутри области действия назад и вперед. Этот случай позволяет вам подражать этому типу деятельности в интерактивном мультике, вызывая скрипт каждый раз, когда мышь попадает по той же самой кнопке кино или МС, в то время как кнопка мыши остается нажатой (рисунок 5).

Прекращение перетаскивания: on (dragOut)

Это событие позволяет вам подражать тому, что может случиться, когда вы нажимаете или касаетесь некоторого объекта, но в случае, если вы коснулись чего-то случайно, а затем убираете вашу руку. Вы можете использовать этот обработчик случая, чтобы вызвать скрипт, когда пользователь размещает указатель поверх кнопки «кино» или МС, нажимает кнопку мыши и перемещает курсор (рисунок 6).

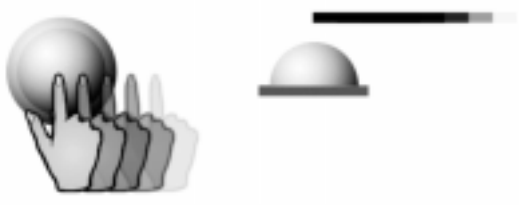


Рисунок 4.

Рисунок 5.

Рисунок 6.

В следующем упражнении мы создадим сканер отпечатка пальца, который взаимодействует с пользователем различными способами. В процессе работы вы примените несколько обработчиков событий.

Создайте новый файл и в этом файле создайте следующие элементы: 4 новых слоя и назовите их следующим образом: нижний слой, который существовал у вас с начала работы с файлом: **background**, следующие слои: **scanner**, **scanner button**, **hand**, **actions**. В слое с фоном создайте фон. Поскольку речь в дальнейшем пойдет о карточной игре, то желательно, чтобы цвет фона был зеленый. В слое с именем **Scanner** создайте MC с **Instance Name** (именем экземпляра) – **scanner_mc.** В случае нашего примера изначально это будет черный квадрат. Войдите в режим редактирования данного MC и в слой, в котором находится черный квадрат, назовите его достаточно традиционно – **background**, далее желательно создать еще несколько слоев: **text**, **Actions**, **labels**. Для красивого визуального эффекта можно создать 1–2 слоя, в которые следует поместить анимацию (рисунок 7).

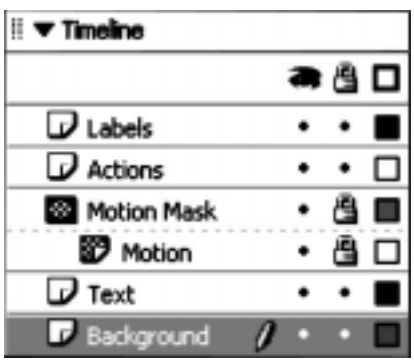


Рисунок 7.

В слое **actions** запишите команду `stop();`, чтобы MC не проигрывался автоматически после загрузки. В слое **Labels** создайте дополнительно 5 ключевых кадров, например, в 10, 30, 49, 80, 150 кадрах и в только что созданных ключевых кадрах задайте метки кадров, чтобы при необходимости было легче ориентироваться при создании проекта и переходить в нужный ключевой кадр. Метки кадров следующие (начиная с первого ключевого кадра): **Off**, **Active**, **Inactive**, **Scan**, **Process**, **Error** (рисунок 8).

Далее в слое **actions** создайте еще несколько ключевых кадров, в которые запишите следующие команды:

29 – в этом ключевом кадре запишите команду `stop()`;

48 – `gotoAndStop("Off")` – то есть после данного кадра MC переедет в ключевой кадр с меткой **Off**;

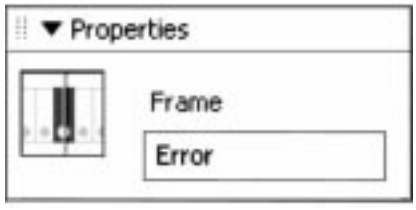


Рисунок 8.

79 – `gotoAndStop("Scan");`
149 – `_root.gotoAndStop("Room")` – вы наверно понимаете, что в данном случае речь идет уже о верхнем уровне, в котором будет ключевой кадр **Room**.

И в самом последнем кадре, уже после того, как пройдет кадр с меткой **Error** (например, в 180 кадре), – `gotoAndStop("Off")`.

В тех же кадрах создайте ключевые кадры в слое **Text** и введите в ключевые кадры фразы по метке того ключевого кадра, в котором находится данный ключевой кадр.

Вернитесь на сцену. В слое **Scanner** создайте текстовое поле, в которое можно будет выводить сообщение о том или ином действии, которое потребуется от пользователя. Текстовое поле может иметь атрибуты либо **Input text**, либо **Dynamic text** и имеет **Instance name** – message_txt (рисунок 9).

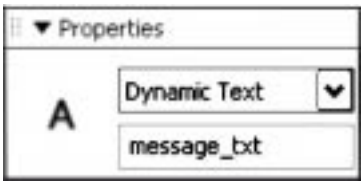


Рисунок 9.

В слое **Scanner button** создайте кнопку, которая по координатам будет совпадать с МС – **scanner_mc** и будет чуть больше самого сканнера (лучше полупрозрачная или прозрачная). Пока скрипты трогать не будем. В слое Hand создайте МС в виде руки и задайте ей имя hand_mc (рисунок 10).

Создадим еще одну сцену, в которой будет находиться игральный стол. Создадим слои **Wood**, **card**, **hand**, **Invisible button**, **Actions**, **Labels**, нижний слой назовем **background**. Слой **background** ничем не отличается от предыдущей сцены. В слое **wood** нарисуем два элемента, имитирующие дерево, на которое надо будет перетаскивать карты. Создадим в том же слое еще одну маленькую «деревянную» панель, преобразуем ее в МС и дадим имя – bump_mc.

В слое **card** нарисуем или импортируем карту, преобразуем ее в МС и назовем – card_mc.

В слое **hand** нарисуем руку, преобразуем ее в МС с именем hand_mc.

В слое **invisible button** создадим МС, желательно пустой и вынесем его за рамки сцены.

В слое **Wood** создадим игральный стол, например, в виде прямоугольника и

на этом прямоугольнике создадим примерно в 20×20 пикселей элемент, преобразуем его в МС и назовем bump_mc.

В слое **labels** зададим метку **Room** (чуть выше мы уже к ней обращались).

На первый взгляд, вся подготовительная работа закончена (рисунок 11), можно приступать к работе.

Вернемся на первую сцену. Выделите уровень кнопки Scanner и затем дважды щелкните по образцу МС сканера в се-

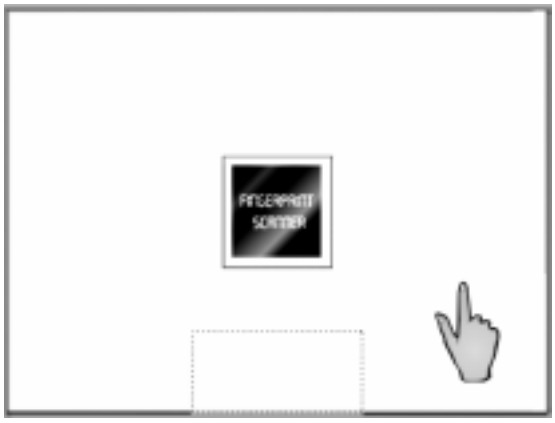


Рисунок 10.

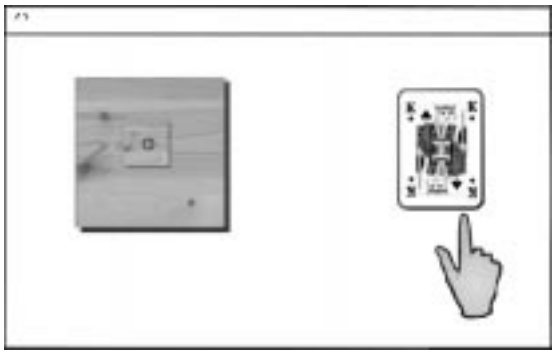


Рисунок 11.

редине сцены, чтобы редактировать его. В слое Actions добавьте следующий скрипт:

```
stop ();
Mouse.hide();
startDrag ("hand_mc", true);
message_txt.text = "Поместите ваш палец на сканер и нажмите. Отпустите через 2 секунды.";
```

Первая команда препятствует приложению уходить с этой сцены, пока не будет дано соответствующее указание. Следующая команда скрывает курсор мыши, потому что мы будем использовать нарисованную руку как собственный курсор. Если посмотреть внимательно на эту команду, то можно увидеть, что к объекту Mouse применяется метод **hide();** следующая команда инструктирует программу перетаскивания MC с именем hand_mc таким образом, как будто это был бы нормальный курсор мыши. Последнее действие размещает указанный текст в текстовом поле message_txt, которое было отведено для сообщений.

Рисунок 12.

Так как этот скрипт находится в первом кадре из первой сцены, он происходит, как только приложение будет загружено и первый кадр отыграет.

С открытой панелью **Actions**, выберите уровень кнопки Scanner, выберите кнопку, которая находится на том же уровне, и добавьте следующий скрипт (рисунок 12):

```
on (rollOver) {
scanner.gotoAndPlay ("Active");
message.text = "Нажмите на экране для продолжения работы. ";
}
```

Когда рука перемещается поверх кнопки, желательно создать впечатление, что она «ощущается» сканером. Случай одновременного

нажатия клавиш позволяет вам делать это, вызывая действие, когда рука сначала находится над сканером, но еще не сделаны действия, позволяющие сканеру проверить вашу руку на идентичность. Этот обработчик действия вызывает две команды: осуществляет переход внутри MC сканера к ключевому кадру с меткой Active, где короткая анимация обеспечивает визуальную команду вызова программы о том, что сканер активен и готов быть нажатым. Тогда изменяется текст, отображаемый в поле текста сообщения.

Добавьте следующий скрипт только ниже текущего скрипта:

```
on (rollOut) {
scanner.gotoAndPlay ("Inactive");
message.text = "Поместите ваш палец на сканер и нажмите. Отпустите через 2 секунды.";
}
```

Как вы можете видеть, одна кнопка может реагировать на несколько событий. Этот скрипт выполняется в том случае, когда мышь сдвигается с кнопки, не нажимая ее. В этом случае переход внутри МС сканера осуществляется на ключевой кадр с меткой **Inactive**, где короткая анимация обеспечивает визуализацию сброса команды сканером. Следующее действие в этом скрипте изменяет текст, отображенный в поле текстового сообщения. Это – тот же самый текст, который появляется, когда приложение только что запустилось, создавая впечатление, что сканер был сброшен.

Ниже добавьте еще один скрипт:

```
on (press) {
scanner.gotoAndPlay ("Scan");
message.text = "Идет сканиро-
вание...";
}
```

Когда пользователь нажимает кнопку, этот скрипт вызывается. Первая команда переводит МС к фрейму с меткой **Scan**. Затем идет команда, изменяющая текст, отображенный в поле текста сообщений.

Добавьте скрипт, только ниже:

```
on (dragOut) {
scanner.gotoAndPlay ("Error");
message.text = "Ошибка. Вы сдвинули
палец во время сканирования. Повторите
сканирование.";
}
```

Этот скрипт с использованием обработчика событий **dragOut**, который вызывает его, когда кнопка мыши нажата, в то время, как пользователь переместился при сохранении нажатой ЛКМ. Суть здесь в том, что пользователь нажимал кнопку для сканирования, но сдвинул палец, что вызвало ошибку в работе сканера и привело к резкому окончанию сканирования. В результате МС сканера переходит к кадру с меткой error, где короткая анимация обеспечивает визуализацию запуска специальной программы в случае ошибки сканирования.

Следующие команды в этом скрипте изменяют текст, отображенный в поле текста сообщений. Добавьте следующий скрипт только ниже текущего скрипта:

```
on (release) {
scanner.gotoAndPlay ("Process");
message.text = "Процедура сканиро-
вания...";
}
```

Когда пользователь отжимает кнопку, вызывается этот скрипт. Первое действие вызывает переход МС сканера к кадру, помеченному **Process**, где короткая анимация указывает, что есть информация для обработки, которая обрабатывается и затем очищается. После прохождения анимации идет команда на переход во вторую сцену, в которой и происходит карточная игра. Эти функциональные возможности позволяют на панели **Timeline** МС перемещаться между сценами. Следующее действие в этом скрипте изменяет текст, отображенный в поле текста сообщения.

Вы теперь закончили работу с первой сценой. В течение воспроизведения с надлежащим пользовательским взаимодействием наш проект должен перейти к сцене **Playroom**. Продолжите работу там.

В панели **Scene** щелкните на сцене с названием **Playroom**.

В этой сцене мы прикрепим события от нажатия мыши к МС. Чтобы облегчить работу, можете использовать события от нажатия мыши почти одним и тем же способом как кнопки, так и МС. В верхней левой части рабочей области вы ранее расположили рядом со сценой пустой МС, который виден как маленький круг. Прикрепите несколько событий от нажатия левой клавиши мыши к этому МС. В середине игрального стола был создан меньший по размеру МС, названный bump_mc. События от нажатия мыши будут размещены и на этом МС. Скрипт будет напи-

сан таким образом, чтобы «зафиксировать», когда им щелкают и перемещают.

Выберите первый кадр из слоя **Actions**, раскройте панель **Actions** и добавьте следующий скрипт:

```
startDrag ("hand_mc", true);
```

Это действие, такое же, как и в предыдущей сцене, дает возможность МС с именем hand_mc, который, как вы помните, представляет собой руку, передвигаться по сцене вместе с курсором мышки. Хотя мышь остается скрытой, когда произойдет переход с одной сцены на другую, перемещение объектов должно быть повторно инициализировано всякий раз, когда сцена изменяется.

Далее, выберите маленький МС с именем bump_mc и добавьте следующий скрипт:

```
on (dragOver) {
this._alpha = this._alpha - 10;
}
```

Данный скрипт говорит о том, что если пользователь нажмет и будет удерживать кнопку мыши при перемещении курсора назад и вперед по этому объекту, то его непрозрачность уменьшится на 10 процентов с каждым проходом (**dragOver**). После 10 проходов bump_mc будет невидим.

С открытой панелью **Actions** выберите МС, представляющий из себя карту, и добавьте следующий скрипт:

```
on (releaseOutside) {
this._x = _root._xmouse;
this._y = _root._ymouse;
}
```

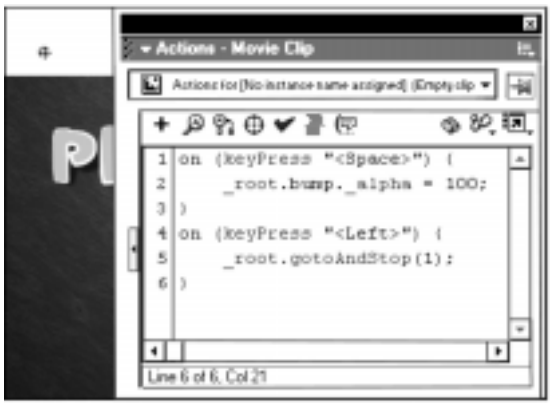


Рисунок 13.

Если мышь помещена на карту, то, нажимая и затем отпуская ЛКМ вне карты, этот скрипт мы запустим. Скрипт будет перемещать карту в те координаты *x* и *y*, в которых будет находиться курсор мыши (точнее, рука, поскольку курсор мыши будет не виден). Другими словами,

предположите, что МС в виде игральной карты находится на левой стороне сцены: если пользователь размещает мышь на карте, нажимает и удерживает кнопку мыши, перетаскивает курсор далеко от нее и затем отпускает кнопку мыши, позиция игральной карты зафиксируется на правой стороне сцены – местоположение курсора мыши в момент отпускания ее левой клавиши.

Выберите пустой МС, который находится выше сцены, и добавьте следующий скрипт (рисунок 13):

```
on (keyPress "<Space>") {
_root.bump._alpha = 100;
}
on (keyPress "<Left>") {
_root.gotoAndStop (1);
}
```

Здесь добавились две команды события, подаваемые с клавиатуры **keyPress** к этому МС. Обратите внимание, что вы не должны помещать МС в видимую часть сцены, потому что команды, записанные на нем для исполнения, должны быть вызваны нажатием клавиш на клавиатуре, а не взаимодействием с курсором.

Первый обработчик **keyPress** будет вызван, когда пользователь нажимает клавишу «пробел». Когда это происходит, прозрачность МС, находящегося на игральном столе будет сброшена к первоначальному значению – 100, снова делая его полностью видимым.

Второй обработчик события **keyPress** будет вызван, если пользователь нажимает стрелку «влево». Когда это произойдет, **Timeline** (временная линейка) вашего приложения (`_root`) перейдет назад, где содержится первая сцена, где пользователь должен отсканировать свой палец снова, чтобы возвратиться к игре.

Это завершает создание нашего проекта. В меню выбираем **Control → Test Movie,** чтобы видеть проект в действии. Данный пример со сканером и затем с элементами игры предназначен для того, чтобы обеспечить ваше понимание, как происходит работа событий во Flash.

Материалы, представленные в статье, используются автором в учебном процессе Санкт-Петербургского регионального центра Федерации Интернет-Образования.

Штенников Дмитрий Геннадьевич,
доцент кафедры физики
СПбГУ ИТМО.