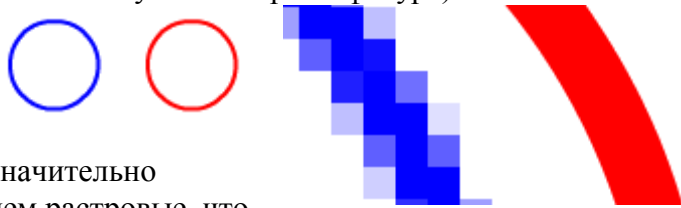


Тема 1. Знакомство с Flash

1. Что такое Flash?

Flash — это технология создания *двухмерной* анимации (т.е. анимации на плоскости). В ней используется главным образом *векторная графика*, т.е. изображения строятся из отдельных геометрических фигур (отрезков, кривых, прямоугольников, окружностей).

В отличие от растровых (точечных) рисунков, **векторные рисунки** описываются математическими формулами и поэтому **не искажаются** при изменении размеров. На рисунке слева показаны растровая и векторная окружности одинакового размера, а на рисунке справа — части этих окружностей при 16-кратном увеличении (красные линии соответствуют векторной фигуре).



значительно
чем растровые, что

на Web-страницах. Правда, для просмотра таких роликов необходимо установить специальную программу-проигрыватель, которая может быть свободно скачана с сайта фирмы *Adobe*.

Векторные рисунки требуют
меньше места в памяти для хранения,
позволяет использовать Flash-анимацию

Технология *Flash* позволяет

- создавать анимированные изображения;
- строить *интерактивные* ролики, реагирующие на действия пользователя;
- строить сложные динамические меню;
- проигрывать видеофильмы.

В то же время она обладает серьезными **недостатками**

- увеличивается размер Web-страницы, что часто неоправданно;
- при проигрывании *Flash*-роликов сильно загружается процессор;
- содержание *Flash*-роликов не доступно для поисковых систем;
- не решена проблема анимации объемных фигур.

История *Flash* началась в 1996 году, когда компания *Macromedia* выпустила продукт под названием *Flash*. В 2005 г. фирма *Adobe* купила *Macromedia* вместе с ее продуктами, включая *Flash*. Последняя на сегодняшний день версия программы называется *Adobe Flash9 (CS3)*. На рисунке слева показан логотип *Flash* фирмы *Macromedia*, а справа — новый логотип фирмы *Adobe*.

Macromedia Flash



Adobe Flash



Далее мы будем рассматривать версию *Adobe Flash9 (CS3)*, однако практически все описанные приемы работают и в версии *Macromedia Flash8*. Главная новинка версии 9 — язык создания сценариев *ActionScript 3.0*, но он непрост для начального уровня и мы не будем его изучать.

2. Форматы файлов

С технологией *Flash* связано несколько форматов файлов. Наиболее известны три из них.

- Файлы с расширением **.FLA** (*FLash Animation*) — это исходные файлы *Flash*-роликов, их можно редактировать. Для размещения в Интернете их надо «опубликовать», превратив в **SWF**-файл.
- Файлы с расширением **.SWF** (*ShockWave Flash*) готовы к размещению в Интернете. Их нельзя изменить в редакторе. Существующие программы для преобразования SWF-файлов в FLA-файлы не позволяют полностью восстановить исходный файл ролика. Кроме того, такое преобразование нарушает права автора.

- Файлы с расширением **.FLV** (*FLash Video*) представляют собой видеоролики, которые можно проигрывать на Web-страницах с помощью специального *Flash*-проигрывателя.

3. Стартовое окно

После запуска программы на экране появляется стартовое меню.



В нем три колонки?

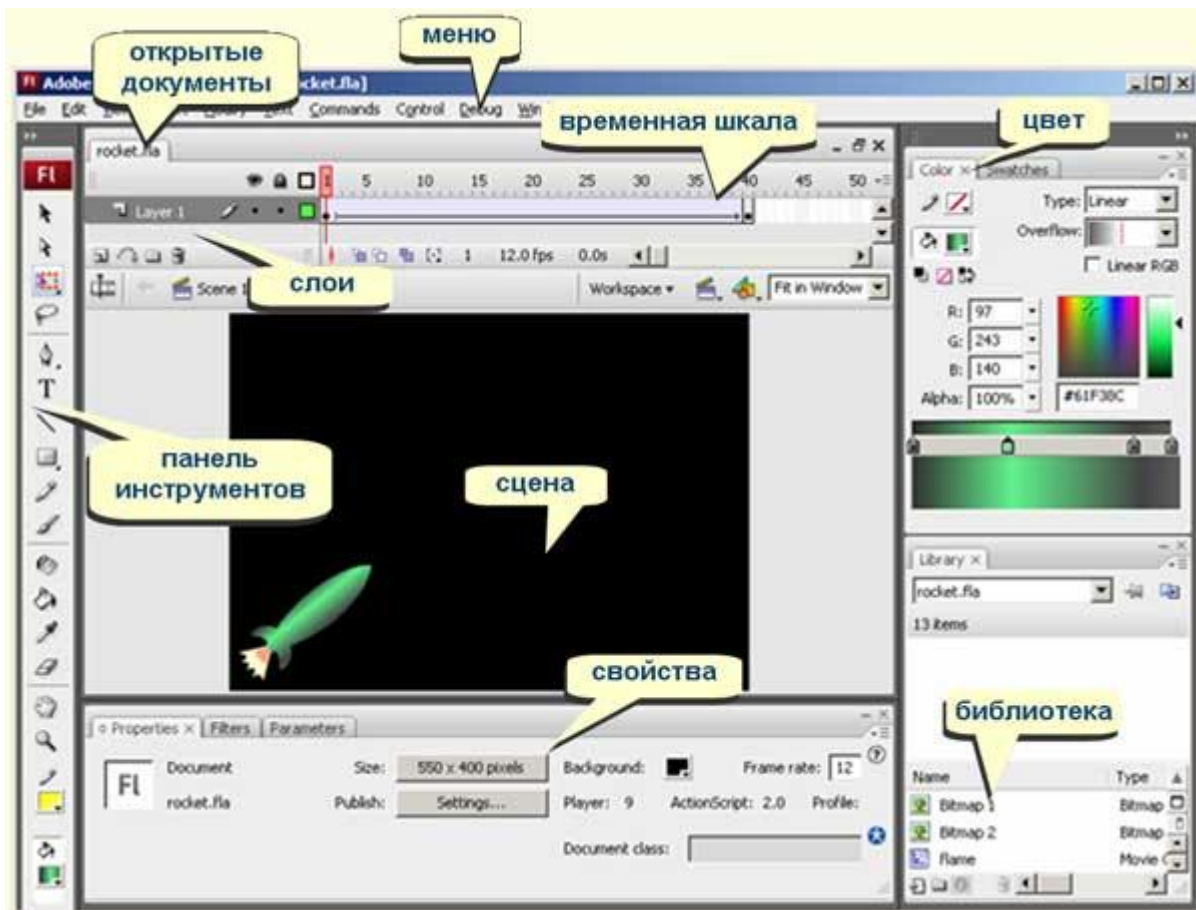
- **Open a Recent Item** — открыть один из последних документов. Вариант *Open...* позволяет выбрать файл на диске.
- **Create New** — создать новый документ. Мы будем создавать документы типа *Flash File (ActionScript 2.0)* — анимацию с поддержкой языка программирования *ActionScript 2.0*.
- **Create from Template** — создать документ стандартного типа из шаблона, например, рекламный баннер (*Advertising*).

Файловые операции выполняются также, как и в других программах, поэтому мы на них не останавливаемся. Отметим только, что пункт меню **Файл** — **Open Recent** позволяет открыть недавно использовавшийся файл, выбрав его из списка.

4. Исследуем готовый фильм

Главное окно


Практика: Запустите программу *Adobe Flash CS3* и откройте файл PRACTICE\1\rocket fla. Затем установите стандартное расположение инструментов, выбрав пункт меню **Window** — **Workspace** — **Default** (*default* — по умолчанию).



В центре окна расположена **сцена** — поле, на котором происходит действие фильма. На рисунке сцена имеет черный цвет, для нового документа — обычно белый. Если разместить объект вне сцены (на сером фоне), он не будет виден, но может потом появиться в кадре, когда будет нужен.

С помощью элементов панели **Properties** (Свойства), расположенной ниже сцены, можно изменить свойства *выделенного объекта*. После загрузки фильма ничего не выделено, поэтому мы видим свойства всего **документа**:

- размер поля (**Size**) — 550 на 400 пикселей;
- цвет фона (**Background**) — черный;
- скорость проигрывания (**Frame rate**) — число кадров, которые сменяются за 1 секунду ($fps = frames\ per\ second$).

Практика : Измените цвет фона на синий с помощью кнопки  справа от надписи **Background**.

Важно : Для быстрого вызова панели **Properties** достаточно нажать клавиши **Ctrl+F3**.


Важно : Комбинация **Ctrl+J** выводит на экран окно свойств документа (меню **Modify-Document**).


Слева от сцены расположена **панель инструментов**, которая содержит инструменты для рисования, выделения и изменения свойств векторных фигур. Справа находятся дополнительные панели: библиотека объектов (**Library**), выбор цвета (**Color**) и другие.

Границы между панелями можно **перетаскивать мышкой**. Нажав кнопку **F4**, можно максимально освободить место для сцены, убрав все панели. Повторное нажатие **F4** восстанавливает стандартный режим.

Временная шкала

Над сценой расположена временная шкала (**Timeline**). Числа над шкалой обозначают номера кадров (в этом фильме 40 рабочих кадров).

Кнопка  на границе между временной шкалой и сценой позволяет убрать с экрана временную шкалу и включить ее снова.

Некоторые кадры отмечены точками . Они называются **ключевыми** — в них происходят существенные изменения. В остальных кадрах (промежуточных) изображение не меняется

или строится программой автоматически. Например, здесь ключевыми являются 1-ый и 40-ой кадры (начальное и конечное положения ракеты), все кадры между ними — промежуточные, путь ракеты рассчитан автоматически.

Красный прямоугольник **1** над линейкой кадров — это **считывающая головка**. Кадр, над которым она находится, изображается на сцене. Головку можно перетаскивать мышкой, таким образом проигрывая фильм вручную.

Практика : Перетащите считывающую головку с 1 до 40 кадра и обратно, посмотрите, как движется ракета.

На временной шкале может быть несколько линеек с кадрами. Это значит, что в фильме есть несколько **слоев**. Для каждого слоя может строиться собственная анимация. Удобно каждый объект размещать на своем слое. Имена слоев показаны слева от временной шкалы. В данном документе есть всего один слой **Layer 1**.

Программа позволяет открывать одновременно несколько файлов. Их имена видны на ярлычках над временной шкалой:



Для перехода к нужному файлу достаточно щелкнуть мышью на его имени. Звездочка справа от имени файла обозначает, что в нем были сделаны изменения,

которые не сохранены на диске.

Просмотр фильма

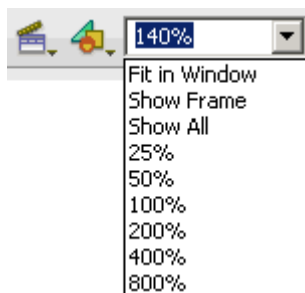
Для быстрого просмотра фильма «на месте» можно нажать клавишу **Enter**, однако при этом не все эффекты анимации будут работать.

Для того, чтобы просмотреть фильма **со всеми эффектами**, надо нажать **Ctrl+Enter** или выбрать пункт меню **Control — Test movie**. При этом фильм откроется в отдельном окне.

Практика : Просмотрите фильм, нажав сначала клавишу **Enter**, а затем — комбинацию **Ctrl+Enter**. Какие **эффекты анимации не работали в первом случае**.

Практика : Измените частоту смены кадров (**Frame rate** на панели **Properties**) на 25 и снова посмотрите фильм. Почувствуйте разницу.

Изменение масштаба



Небольшое окно над правым верхним углом сцены показывает масштаб изображения. Вариант **Show Frame** регулирует размеры так, чтобы все объекты данного кадра были видны, при выборе **Fit in Window** сцена «вписывается» в оставшееся свободное место при каждом изменении размеров окна. **Show All** означает «показать все объекты».

Изменить масштаб можно также с помощью меню **View — Magnification** или включив инструмент **Zoom** (Масштаб) на панели инструментов. При использовании инструмента **Zoom** щелчок мышью приводит к увеличению изображения в 2 раза, центр видимой части перемещается в точку щелчка. Для уменьшения изображения надо при щелчке удерживать клавишу **Alt**.

При больших увеличениях часто надо переместить «окно» (видимую часть изображения) в нужное место. Для этого можно использовать линейки прокрутки или инструмент **Hand** (Рука), который позволяет «схватить и перетащить» изображение в нужную точку.

Инструмент **Hand** можно включить временно, удерживая нажатой клавишу «**пробел**».

Практика : Установите масштаб 400% и с помощью инструмента **Hand** (Рука) перейдите к области с ракетой. Затем включите инструмент **Zoom** и попробуйте его в работе в режимах увеличения и уменьшения.

Важно : Для временного включения инструмента **Zoom** в режиме увеличения можно нажать **Ctrl+пробел**, а для уменьшения — **Ctrl+Alt+пробел**. Кроме того, комбинации **Ctrl+плюс** и **Ctrl+минус** соответственно увеличивают и уменьшают масштаб в 2 раза.

Публикация и экспорт

Публикация — это сохранение фильма в формате **SWF** и создание Web-страницы (файла на языке HTML), в которую встроены наш ролик.

Практика : Сохраните фильм в виде Web-страницы, выбрав пункт меню **File—Publish** или нажав комбинацию клавиш **Shift+F12**. В программе **Проводник** зайдите в папку PRACTICE\1 и откройте файл rocket.html. Вы должны увидеть Web-страницу, на которой проигрывается ролик.

Экспортом называется сохранение фильма в каком-либо другом формате, который не является «родным» для среды *Flash*. Для экспорта фильма надо выбрать пункт меню **File—Export—Export Movie**. После этого появится диалоговое окно сохранения файла. В поле **Тип файла** нужно установить нужный формат и ввести имя файла (или оставить имя по умолчанию). На экран будет выведено окно с параметрами сохранения (здесь можно поменять размер окна и установить желаемое качество).

Практика : Сохраните ролик в формате **Animated GIF** (GIF-файл с анимацией). Параметры сохранения, предложенные программой, оставьте без изменений. С помощью **Проводника** просмотрите полученный файл.

Практика : Сохраните ролик как видеофайл в формате **Windows AVI**. Параметры сохранения, предложенные программой, оставьте без изменений. В следующем окне выберите программу для сжатия **Microsoft Video** и щелкните по кнопке **OK**. С помощью **Проводника** определите и сравните размеры файлов в форматах **SWF**, **GIF** и **AVI**. Просмотрите полученные файлы и сравните их качество.

Вопрос для самостоятельной работы.

1. Что такое Flash?
2. Возможности технологии *Flash* .
3. Недостатки технологии *Flash*.
4. Форматы файлов, связанные с технологией *Flash*.
5. Опишите главное окно программы.
6. Для чего предназначена панель инструментов.
7. Что такое временная шкала.
8. Как запустить просмотр со всеми эффектами.




Термины: Flash, расширения файлов Flash, сцена, публикация, экспорт файла.

Литература: [1 С.28-58, С.554-614 ; 2 С.47-58]

Тема 2. Контуры

1. Цвета



Векторный рисунок в программе *Flash* состоит из отдельных элементов, которые задаются узловыми точками (*anchor points*). Это могут быть отрезки, ломаные, прямоугольники, овалы и др. Некоторые из этих инструментов

 **Line** — линия  **Pencil** — карандаш  **Pen** — перо





рисуют только **контуры** (*stroke*), другие

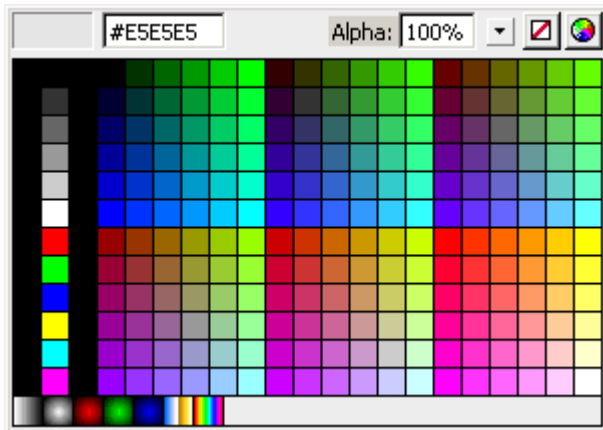
 **Brush** — кисть

создают только области (**заливки**, *fill*) без контура, а третьи



 **Rectangle** — прямоугольник  **Oval** — овал

имеют контур и заливку.

Цвет контура можно выбрать с помощью кнопки   на панели инструментов, а цвет заливки — с помощью кнопки  . При этом открывается окно с палитрой:






В верхнем левом углу палитры показан **код цвета** (в шестнадцатеричной системе). Параметр **Alpha** управляет прозрачностью: при $Alpha=100\%$ цвет полностью непрозрачен, при $Alpha=0\%$ — полностью прозрачен.

Кнопка  в правом верхнем углу палитры позволяет выбрать прозрачный цвет (у фигуры не будет контура или заливки), а кнопка  вызывает стандартный диалог выбора цвета (если цветов, включенных в палитру, не хватает).






В нижней строчке палитры можно выбрать **градиентные заливки**. Далее мы будем сами строить такие заливки с помощью панели **Colors** (она расположена в правом верхнем углу экрана).

Ниже кнопок выбора цвета в палитре инструментов находятся еще три вспомогательные кнопки:  — установить черный контур и белый цвет заливки;

 — поменять цвета контура и заливки;  — выбрать прозрачный цвет (нет контура или заливки).

2. Линии

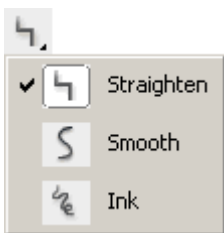
Для рисования линий предназначены три инструмента:

 **Line** — линия  **Pencil** — карандаш  **Pen** — перо

Карандаш

Карандаш рисует так же, как и в редакторе *Paint*.

Практика : Запустите программу *Flash*, создайте новый документ и сохраните его в папке PRACTICE\2. Выберите цвет линий, включите карандаш и нарисуйте от руки квадрат.



Наверняка вы заметили, что стороны квадрата оказались ровными, несмотря на то, что вы провели их несколько кривовато. Дело в том, что *Flash* пытается «улучшить» линию, а метод «улучшения» выбирается из дополнительного меню **Options** (режимы, настройки), которое появляется в нижней части панели инструментов. Меню для инструмента **Pencil** (см. рисунок справа) содержит 3 режима:

- **Straighten** — выпрямлять;
- **Smooth** — сглаживать;
- **Ink** — не изменять.

Сейчас включен вариант **Straighten**, это объясняет полученный эффект спрямления сторон квадрата.

Практика : Включите режим **Smooth** и нарисуйте от руки окружность. Затем выберите режим **Ink** и нарисуйте еще одну окружность, сравните результаты.

В меню **Modify-Shape** есть команды **Straighten**, **Smooth** и **Optimize**, предназначенные для обработки уже нарисованных контуров:

- **Straighten** — выпрямить, линия становится более угловатой;
- **Smooth** — сгладить углы;
- **Optimize** — попытаться уменьшить количество узлов.


Практика : Нарисуйте линию в режиме **Ink**, выделите ее и попробуйте несколько раз применить метод **Straighten**, а затем — несколько раз метод **Smooth**.

Метод **Optimize** позволяет уменьшить количество узлов линии. Это очень важно, потому что чем больше узлов, тем больше времени требуется на рисование линии, особенно при анимации.

При выборе пункта меню **Modify-Shape-Optimize** появляется окно точной настройки. По окончании действия выводится сообщение о результатах: сколько было сегментов (участков) в исходной кривой (*original shape*), сколько осталось (*optimized shape*) и каков процент упрощения (*reduction*).

Практика : Измените цвет карандаша, нарисуйте в режиме **Ink** какую-нибудь сложную линию. Выделите ее двойным щелчком при нажатой клавише **Ctrl** и примените к линии метод **Modify-Shape-Optimize** и сравните, сколько узлов было и сколько осталось.

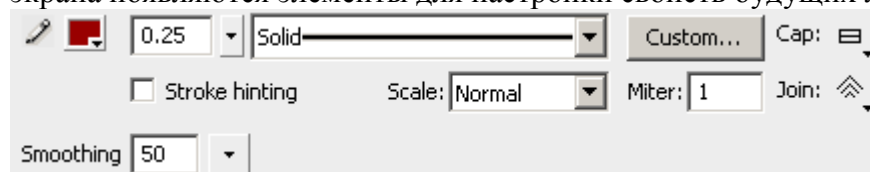
Линия

Инструмент  **Line** (линия) работает так же, как и в реакторе *Paint* — рисует отрезки выбранного цвета.

Практика : Включите инструмент **Line** и нарисуйте несколько отрезков разных цветов.

Параметры

При включении инструмента **Pencil** или **Line** на панели **Properties** (свойства) в нижней части экрана появляются элементы для настройки свойств будущих линий



В левом верхнем углу — кнопка для выбора **цвета** линии, такая же, как и на панели инструментов. Правее — окно для выбора **толщины** линии (сейчас установлена толщина 0,25), при щелчке на стрелке справа появляется движок, которым можно изменить значение. Еще правее — список для выбора **стиля** линии (сейчас выбран вариант *Solid* — сплошная линия). Кнопка **Custom** позволяет создать новый стиль линии. Остальные параметры:

- **Scale** определяет, как будут меняться размеры объекта при изменении размеров фильма;
- **Stroke hinting** позволяет улучшить качество скругленных участков для мелких элементов;
- **Caps** определяет стиль конечных точек линий (резкий обрыв, скругление, дорисовать квадрат). Разница видна только при большом увеличении или для толстых линий:

Cap




Join задает стиль углов:

Join




- В первом случае параметр **Miter** в отдельном окне определяет остроту углов.
- **Smoothing** определяет степень сглаживания для инструмента **Pencil** в режиме сглаживания.

3. Выделение объектов


Если изменить параметры на панели **Properties**, новые значения будут влиять на новые линии. Чтобы изменить свойства существующей линии, ее нужно сначала выделить. Для этого используется инструмент  **Selection** (выделение, стрелка).

Важно : Инструмент  можно быстро включить, нажав на клавишу **V**.

Практика : Включите инструмент  **Selection** и щелкните по одной из сторон квадрата. Вы увидите, что выделена только одна сторона. Теперь можно отдельно установить свойства этого куска и даже переместить его в другое место, просто перетащив мышкой.


Для того, чтобы выделить **всю линию**, нужно сделать на ней *двойной щелчок*. После этого ее свойства можно изменить с помощью панели **Properties**.

Практика : Выделите весь квадрат, измените цвет его контура и установите толщину линии 5.

Существует еще один способ выделения: включить инструмент  **Selection** и обвести все нужные объекты в прямоугольник.

4. Перекраска контура

Инструмент *Ink Bottle*


Для изменения цвета можно также использовать инструмент  **Ink Bottle** (чернильница). Включив этот инструмент, нужно установить желаемые свойства линии на панели **Properties** и щелкнуть мышью на нужном контуре.

Практика : С помощью инструмента **Ink Bottle** перекрасьте один из нарисованных контуров.

Отмена операции

Для того, чтобы отменить сделанную операцию, надо нажать клавиши **Ctrl+Z** или выбрать пункт меню **Edit—Undo...** (после слова *Undo* стоит название отменяемой операции). По умолчанию программа помнит 100 действий пользователя.

Если вы хотите вернуть отмененную операцию, надо нажать **Ctrl+Y** (меню **Edit—Redo...**).

Практика : Выделите все объекты, обведя их в рамку инструментом  **Selection** и нажав на клавишу **Delete**. Затем отмените удаление.

5. Направляющие

Вставка кадра

Мы продолжим рисование на чистом листе. Чтобы не удалять все нарисованное, можно просто перейти на новый кадр. Для этого его надо создать.

Существует два вида кадров: **ключевые** и промежуточные. В ключевых кадрах происходит существенная смена изображения, в промежуточных картинка не изменяется или строится программой автоматически.

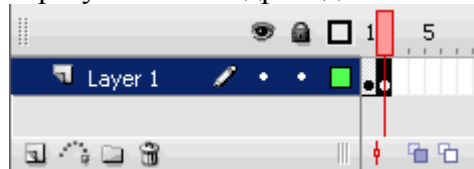
Для вставки кадра выделим щелчком нужный кадр на шкале времени (над сценой) и нажмем одну из клавиш:

- **F5** (меню **Insert—Timeline—Frame**) — промежуточный кадр;
- **F6** (меню **Insert—Timeline—Keyframe**) — ключевой кадр, изображение копируется из предыдущего ключевого кадра;
- **F7** (меню **Insert—Timeline—Blank Keyframe**) — пустой ключевой кадр.

Кроме того, можно использовать **контекстное** (всплывающее меню): щелкнуть правой кнопкой на нужном кадре и выбрать команду **Insert Frame**, **Insert Keyframe** или **Insert Blank Keyframe**.

Практика : Вставьте пустой ключевой кадр в кадр 2.

В результате в кадре 2 должна появиться жирная точка, сцена должна быть пустая.



Направляющие

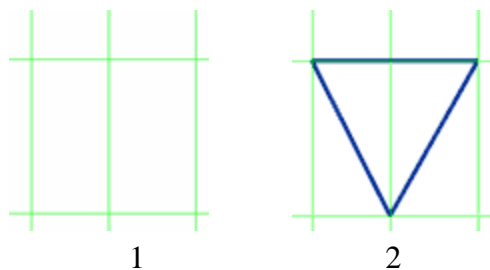
Направляющие — это линии, которые помогают выравнивать элементы рисунка. Чтобы добавить направляющие, включите линейки, выбрав пункт меню **View—Rulers** (на линейках дана разметка в пикселях). Затем надо нажать левую кнопку мыши на линейке и, не отпуская ее, «вытащить» направляющую на сцену.

Чтобы **убрать** направляющую, просто перетащите ее за пределы сцены.

Иногда направляющие надо временно скрыть (убрать с экрана), чтобы они не мешали при работе. Для этого нужно нажать клавиши **Ctrl+;** или выбрать пункт меню **View—Guides—Show Guides**.

Другие операции с направляющими можно найти в меню **View—Guides**.

Практика : Вставьте пустой ключевой кадр в кадр 3. Расставьте направляющие (зеленые линии) так, как показано на рисунке 1.



Для того, чтобы узловые точки линий «прилипали» к направляющим, нужно включить флажок **View—Snapping—Snap to Guides** (в меню).



Практика : Нарисуйте треугольник из трех отрезков, как показано на рисунке справа (вверху).

6. Изменение формы контура

С помощью инструмента  **Selection** можно

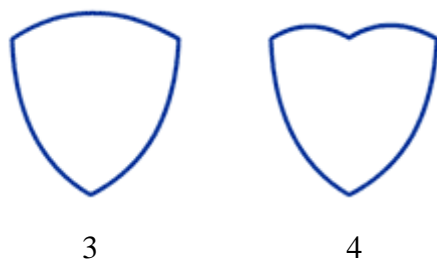
- **переместить** узловые точки контура — просто перетащить мышью;
- **изогнуть** сегмент линии — «схватить» линию между узловыми точками и перетащить в нужное положение;
- **добавить** новую угловую точку — удерживая клавишу **Ctrl**, щелкнуть по линии и перетащить появившуюся новую опорную точку.


Когда курсор мыши может выглядеть по-разному:

-  — над криволинейным участком;
-  — над угловой точкой.

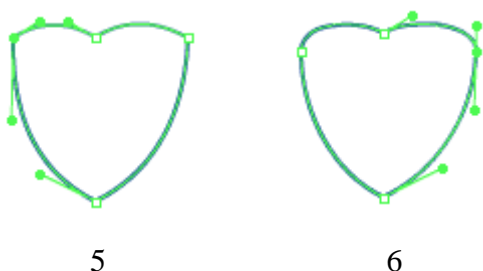
Для выполнения следующих операций удобнее скрыть направляющие (они по-прежнему работают, но не видны). Для этого снимите флажок в меню **View—Guides—Show Guides**.

Практика : Изогните стороны треугольника, как показано на рисунке 3 (ниже). Затем добавьте новую угловую точку в середине верхней дуги и оттяните ее вниз (рисунок 4).




У нас получилась фигура, напоминающая по форме сердце. Остается «скруглить» два угла в верхней части. Это можно сделать с помощью инструмента  **Subselection** (частичное выделение) на панели инструментов (клавиша **A**).


Если щелкнуть этим инструментом по контуру, вы увидите все его узловые точки. Эти точки можно перетаскивать мышкой. Кроме того, если щелкнуть на какой-то опорной точке, появляются **касательные** — отрезки с точками на концах. Они определяют углы выхода и кривизну линий, выходящих из этой точки. Схватив за конец касательной, можно вращать ее и изменять длину.




Если обе касательные находятся на одной прямой, это **гладкий узел**, здесь направление линии меняется плавно. Если надо сделать из гладкого узла угловой, перетаскивайте касательные при нажатой клавише **Alt**.

Практика : Включите инструмент  **Subselection** (клавиша **A**), перетащите верхние угловые точки немного вниз и отрегулируйте их касательные так, как показано на рисунке 6 (сделайте эти точки гладкими узлами и укоротите нижние касательные).


7. Добавление заливки

В результате мы получили замкнутый контур сердечка. К такому контуру можно легко добавить **заливку**. Для этого используется инструмент  **Paint Bucket** (ведро с краской).


Практика : Включите инструмент **Paint Bucket**, установите красный цвет заливки (на панели **Properties** или с помощью кнопки   на панели инструментов) и закрасьте контур, щелкнув внутри.

С помощью инструмента  **Paint Bucket** можно закрашивать контуры, имеющие небольшие разрывы. Величина допустимого разрыва регулируется с помощью дополнительного меню (оно появляется в нижней части панели инструментов при включенном инструменте **Paint Bucket**).



8. Перо

Инструмент  **Pen** (перо) предназначен для рисования сложных контуров, включающих прямолинейные и скругленные участки (*сегменты*).


Для рисования ломаной надо при включенном инструменте **Pen** щелкать мышкой в ее узлах. Если же нужен сглаженный узел, нужно протащить мышью в сторону продолжения линии при нажатой левой кнопке.


Чтобы удалить последний узел, надо нажать клавишу **Backspace**. Двойной щелчок завершает построение незамкнутого контура. Если же надо **замкнуть контур**, последний щелчок выполняется на первом узле, при наведении на него около курсора мыши появляется окружность .


Свойства инструмента **Pen** (на панели **Properties**) такие же, как для инструментов **Line** и **Pencil**.

После того, как контур нарисован, его можно редактировать с помощью инструментов  **Selection** (клавиша **V**) и  **Subselection** (клавиша **A**) так же, как описано выше.

Для изменения контура можно также использовать и сам инструмент **Pen**. В зависимости от положения мыши, курсор может иметь различный вид, показывая, что будет сделано при щелчке в этом месте


 ставим начальный узел контура;

 ставим очередной (не первый) узел контура;

 добавляем узел (мышь наведена на сегмент между узлами);


 удаляем узел (мышь наведена на угловой узел);


 продолжить контур от этой точки (в данный момент не начат ни один контур);

 преобразовать сглаженный узел в угловой.

После включения инструмента **Pen** для редактирования контура нужно щелчком выделить его и использовать описанные выше приемы.

При нажатой клавише **Ctrl** можно **перетаскивать** весь контур (за сегмент между узлами) и отдельные узлы.

Если удерживать клавишу **Alt**, курсор приобретает форму угла . Щелчок по узлу в этом режиме преобразует угловой узел в гладкий. Кроме того, можно изменять касательные гладких узлов так же, как и инструментом **Subselection**.

Практика : Вставьте новый ключевой кадр в кадр 3. Используя инструмент  **Pen**, нарисуйте контур, изображающий профиль морской волны, как показано на рисунке. Закрасьте контур синим цветом с помощью инструмента **Paint Bucket**.



Сейчас у нас три ключевых кадра с разными изображениями. Если просмотреть фильм, нажав на **Ctrl+Enter**, можно увидеть, как они мелькают на экране.

Практика : Включите инструмент **Selection** и щелкните мышью в свободной области (вне объектов), чтобы увидеть свойства документа на панели **Properties**. Установите частоту 1 кадр в секунду и просмотрите ролик.

Практикум «Изменение формы контура»

Рисуем сердце и волну. Рисуем прямоугольник карандашом и делаем из него дельфина.

Вопрос для самостоятельной работы.

1. Какие инструменты рисуют только контуры.
2. Какие инструменты рисуют только заливание.
3. Какие инструменты рисуют контуры и заливания.
4. Какой параметр управляет прозрачностью.
5. Какие виды кадров существуют в Flash.
6. Что делают клавиши F5, F6, F7.

Термины: параметр Alpha, виды кадров, направляющие.

Литература: [1 С.62-86]

Тема 3. Покадровая анимация

3.1. Работа с кадрами

Покадровая анимация состоит в том, что каждый кадр фильма рисуется отдельно, и они быстро сменяют друг друга. При частоте 24 кадра в секунду и выше человек воспринимает такие изменения как непрерывное движение.

Кадры изображаются серыми прямоугольниками на временной шкале. Как вы уже знаете, бывают кадры двух типов:

- **ключевые** — в них происходит существенное изменение изображения;
- **промежуточные** — в них изображение не меняется или строится программой автоматически.

Кадры можно перемещать, копировать и удалять. Для **перемещения** надо выделить кадр щелчком и перетащить его в новое место на временной шкале. На старом месте остается обычный (промежуточный) кадр, то есть изображение будет скопировано из ближайшего слева ключевого кадра.

Если при перетаскивании кадра удерживать клавишу **Alt**, кадр **копируется** в новое место (на это указывает знак «плюс» у курсора).

Для удаления кадра надо выделить его, нажать правую кнопку мыши и выбрать команду **Remove Frames** (удалить кадры) из контекстного меню.

Эти операции можно выполнять и с несколькими выделенными кадрами. Для выделения можно протащить через них мышку при нажатой левой кнопке. Щелчками при нажатой клавише **Ctrl** можно выделить «несоседние» кадры.


Для работы с кадрами можно использовать специальный **буфер обмена** — команды **Cut Frames** (вырезать), **Copy Frames** (копировать) и **Paste Frames** (вставить) из контекстного меню.

Практ: Откройте фильм, созданный на прошлом уроке, и сохраните его с другим именем в папке PRACTICE3 (для этого нужно выбрать пункт меню **File—Save as (Ctrl+Shift+S)**). Скопируйте кадр 2 в кадр 4 и просмотрите фильм. Затем удалите кад 2.

3.2. Трансформации объектов

Практ: Удалите все кадры из фильма.

Если сейчас попытаться что-то нарисовать на сцене, будет выдано сообщение об ошибке, поскольку рисовать можно только в ключевых кадрах, а они все удалены.


Практ: Создайте новый ключевой кадр в кадре 1. Выберите темно-зеленый цвет контура, толщину линии 3. С помощью инструмента  **Pen** нарисуйте контур листа с прожилкой посередине. Залейте его светло-зеленым цветом.

Мы сделаем покадровую анимацию, которая изображает увеличение листа.



Практ: Выделите кадр 1 и вставьте 4 новых ключевых кадра, нажимая **F6**.

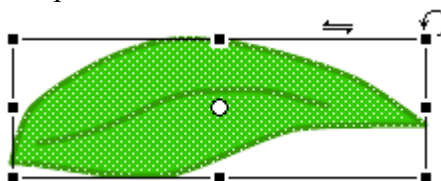
Сейчас во всех кадрах — одинаковое изображение. Мы изменим фильм так, чтобы на первом кадре был самый маленький лист, а потом — все больше и больше.

Практ: Включите инструмент  **Selection** и выберите кадр 1. При этом все его содержимое будет также выделено.

Для уменьшения листа на 1-ом кадре применим инструмент **Free Transform** (свободные преобразования), который включается кнопкой  на панели инструментов, клавишей **Q**, или командой **Free Transform** из контекстного меню (после нажатия правой кнопки мыши).

Вокруг выделенного объекта появляется рамка с маркерами, такая же, как, например, в редакторе *Word*. Теперь можно:

- изменить **размеры**, перетаскивая за маркеры (при нажатой клавише **Shift** пропорции не меняются, при нажатии **Alt** точка вращения остается неподвижной);
- **повернуть** объект вокруг центра, обозначенного белым кружком (для этого нужно подвести мышь к углу рамки снаружи, чтобы курсор принял вид ) , центр вращения можно перетаскивать мышкой;
- **перекосить** объект, схватив и потянув мышкой за рамку между маркерами (курсор ) ;
- **перетащить углы** рамки при нажатой клавише **Ctrl**.






Практ: Уменьшите размеры листа примерно в 3 раза. Перетащите центр вращения в левый нижний угол листа и поверните лист немного более вертикально.

Практ: Таким же способом постройте промежуточные рисунки в кадрах 2-4. Просмотрите фильм.

Другие варианты преобразования применяются с помощью меню **Modify—Transform**. В частности можно

- повернуть объект на 90 градусов (**Rotate 90° CW** — *clockwise*, по часовой стрелке; **Rotate 90° CCW** — *counter-clockwise*, против часовой стрелки);
- построить отражение объекта по вертикали (**Flip Vertical**) или горизонтали (**Flip Horizontal**);
- выполнить тонкую настройку контура с помощью команды **Envelope**.

Когда включен инструмент **Transform**, в нижней части панели инструментов появляются кнопки, позволяющие включать какой-то один вид преобразований:


-  **Rotate and Skew**, вращение и поворот;
-  **Scale**, изменение размеров;
-  **Distort**, искажение, свободное перетаскивание узлов;

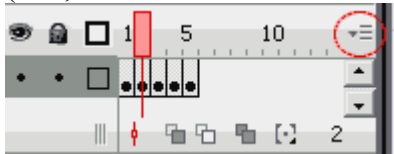
Envelope, работа с огибающей.

При нажатии клавиш **Ctrl+T** открывается панель **Transform**, на которой можно точно задать угол поворота и скоса.

Важно: При всех трансформациях объект не «портится». Если надо отменить все преобразования и вернуть его к исходному виду, выберите пункт меню **Modify—Transform—Remove Transform**

3.3. Меню панели

Далее нам будет удобно сделать кадры на временной шкале пошире. Для этого используется **меню временной шкалы**, которое вызывается щелчком по кнопке  в левом верхнем углу (там, где кончаются числа — номера кадров).



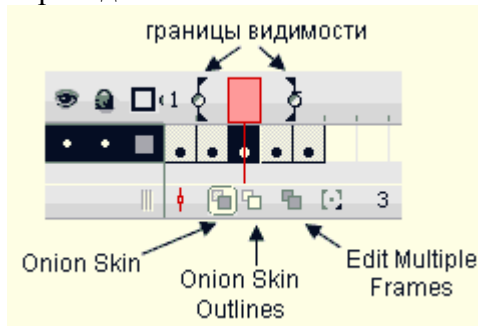
Важно: Аналогичные меню есть у всех панелей *Flash*.

С помощью этого меню можно выбрать ширину кадров на шкале, а также изменить расположение шкалы времени (группа **Placement**) и уменьшить **высоту** кадров (установив режим **Short**).


Практ: С помощью меню временной шкалы установите ширину кадров **Medium** (средний).

3.4. «Луковая кожура»

Скорее всего, анимация с первого раза получится не очень равномерная. Для настройки покадровой анимации в программе *Flash* есть специальное средство, которое называется **Onion Skin** («луковая кожура»). Если включить этот режим, при обработке очередного кадра вы будете видеть полупрозрачные изображения соседних кадров, что поможет улучшить переходы.



Для включения этого режима надо щелкнуть по кнопке **Onion Skin** в нижней части временной шкалы (см. рисунок выше). Если нужно оставить **только контуры** фигур на соседних кадрах (чтобы заливка не мешала), можно щелкнуть по кнопке **Onion Skin Outlines** (контуры «луковой шелухи»). Область видимости регулируется движками над временной шкалой.

Щелкнув по кнопке , вы увидите меню, с помощью которого можно изменить настройки «луковой кожуры»



- всегда выводить маркеры — **Always Show Markers**;
- закрепить маркеры — **Anchor Onion** (иначе они смещаются вслед за читающей головкой);
- выделять 2 или 5 кадров слева от текущего — **Onion 2** или **Onion 5**;
- выделять все кадры — **Onion All**.

Практ: Включите режим **Onion Skin Outlines**, установите видимость только двух соседних кадров. Попробуйте сделать более ровные переходы между кадрами. Проиграйте фильм.

3.5. Редактирование нескольких кадров

Теперь представьте, что нужно переместить этот лист, например, вправо. Придется перемещать его во всех кадрах, которых может быть достаточно много. К счастью, есть более легкий способ.

Справа от кнопки **Onion Skin Outlines** расположена кнопка **Edit Multiple Frames** (редактировать несколько кадров). Она позволяет заменять положение объекта сразу во всех кадрах, ограниченных движками зоны видимости (над временной шкалой).

Практик: Включите режим  **Edit Multiple Frames** и раздвиньте движки видимости так, чтобы они охватывали все кадры с 1-ого по 5-ый. Включите инструмент  **Selection**, выделите рамкой изображение листа на всех кадрах и перетащите справа, чтобы слева можно было поставить еще один лист.

В этом режиме можно выполнять все действия также, как и обычно, разница в том, что захватываются несколько кадров.

Практик: Выделите снова все изображения листа и скопируйте их влево, перетащив при нажатой клавише **Alt**. Постройте отражение нового листа (**Modify—Transform—Flip Horizontal**) и переместите его так, чтобы нижние точки (точки вращения) первого и второго листов совпали. Просмотрите фильм.


3.6.


Практикум «Редактирование нескольких кадров»

Рисуем раскрывающиеся листики и рост стебля с цветком посередине. Можно добавить летящую пчелку или ползущего жука.

Задание на дом – покадровая анимация «Рост цветка и движущийся персонаж» 6-8 кадров.


3.7. Кисть


Инструмент  **Brush** (кисть) предназначен для создания заливок без контура. При включенном инструменте **Brush** в нижней части панели инструментов показаны три кнопки для его настройки (сверху вниз):



 режим работы кисти:

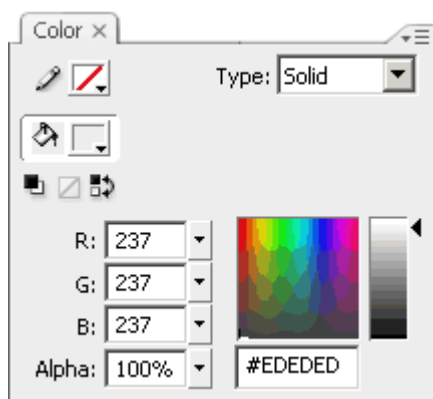
- **Paint Normal** (нормальный) — закрашивается все;
- **Paint Fills** (заливки) — закрашиваются только заливки, контуры не изменяются;
- **Paint Behind** (задний план) — закрашиваются только те области, где нет ни контуров, ни заливок;
- **Paint Selection** (выделение) — закрашивается только та заливка, которая была заранее выделена;
- **Paint Inside** (внутри) — закрашивается только та заливка, откуда началось закрашивание.

 размер кисти;

 форма кисти (если выбрана круглая кисть, то на этой кнопке будет такой же рисунок, как и на предыдущей).


Практика : Откройте файл `brush.fla` из папки `PRACTICE\3`. Используя инструмент  **Paint Bucket**, закрасьте все квадраты разными цветами.

Практика: Выберите фиолетовый цвет заливки и включите инструмент  **Brush** и установите максимальный размер кисти. Включая поочередно все 5 режимов работы кисти, проведите 5 вертикальных линий через квадраты. Для 4-ого режима (**Paint Selection**, закраска выделенной области) нужно сначала выделить заливку 4-ого квадрата инструментом  **Selection**. Для 5-ого режима (**Paint Inside**, рисунок внутри) надо начинать рисование внутри заливки.



8. Панели Color и Swatches


Если нужного цвета нет в палитре, нужно использовать панель **Color** (цвет), расположенную в правом верхнем углу

экрана. Сейчас в левом верхнем углу (см. рисунок) нажата кнопка , это означает, что устанавливается цвет заливки.

Цвет можно выбрать с помощью движков, задать в виде тройки составляющих **RGB** (Red, Green, Blue) или ввести в виде шестнадцатеричного кода (в окне, где сейчас написано **#EDED**).

Параметр **Alpha** обозначает непрозрачность цвета в процентах (от 0% — полностью прозрачный — до 100% — полностью непрозрачный).

Панель **Swatches** (образцы) содержит палитру цветов, доступных при выборе цвета в панели инструментов и панели **Properties**. Можно добавлять цвета в палитру и удалять их.


Для того, чтобы добавить новый цвет в палитру, постройте его на панели **Colors**. Затем откройте меню палитры (щелкнув по кнопке  в правом верхнем углу) и выберите пункт **Add Swatch**.

Чтобы удалить образец из палитры, надо выделить его на панели **Swatches** и выбрать пункт **Delete Swatch** из меню этой палитры.

3.9. Практикум (кисть)

Покадровая анимация Медведя на 6-8 кадров.

История про медведя, который на что-то (например лопается воздушный шар) смотрит и моргает глазами.

Практика : Создайте новый документ и сохраните его в папке PRACTICE\3. Включите инструмент  **Brush** и установите с помощью панели **Color** цвет с параметрами **R=134**, **G=71** и **B=64**. Добавьте этот цвет в палитру.

Мы будем рисовать кистью Винни-Пуха. Вот такого:



Практика : Установите самый большой размер кисти и значение параметра **Smoothing**, равное 50 (на панели **Properties**). Нарисуйте отдельно голову и тело Винни-Пуха (рисунок1).



1



2






3




4

Редактирование заливки

Для редактирования заливки можно использовать инструмент  **Eraser** (ластик, стирательная резинка). Он работает примерно так же, как и кисть. Внизу на панели инструментов можно выбрать размер и форму ластика, кнопка  позволяет установить режим стирания (нормальный, только заливки, только линии, только выделенные заливки, только та заливка, с которой начали).

Если включить кнопку  **Faucet**, одним щелчком стирается сразу вся заливка или весь контур, на котором щелкнули.

Практика : Попробуйте в действии инструмент .


Контуры заливки, нарисованной от руки, можно улучшить с помощью верхнего меню

Modify—Shape. Оно содержит пункты


- **Smooth** — сгладить границу;
- **Straighten** — выпрямить (сделать более угловатой).

Практика : Выделите голову Винни-Пуха и примените сглаживание (**Modify—Shape—Smooth**). Прделайте то же самое с туловищем.

Детали

Практика : помощью инструмента  **Selection** выделить голову и перетащите ее вниз, соединив с телом (рисунок 3). Теперь снимите выделение, щелкнув на пустом месте, и снова выделите голову

Вы наверняка заметили, что выделилась вся фигура, голова вместе с телом (рисунок 4). Это означает, что при соприкосновении (и наложении) заливок одного цвета они **сливаются** в один объект.

Практика : Включите инструмент  **Brush** и установите с помощью панели **Color** цвет с параметрами **R=81, G=41 и B=27**. Добавьте этот цвет в палитру.

Практика : Нарисуйте этой кистью глаз, правое ухо, правую руку и правую ногу Винни-Пуха (рисунок 5).




5

6

7


8


Если теперь с помощью инструмента  **Selection** выделить ухо и перетащите его вверх, вы увидите, что эта область в «нижней» заливке вырезана. Таким образом, верхняя заливка другого цвета **вырезает** область из заливки, оказавшейся внизу.

Практика : Включите снова кисть, установите режим **Paint Behind**, чтобы кисть не затрагивала уже существующие фигуры, и нарисуйте левое ухо, левую руку и левую ногу (рисунок 6).

Практика : Переключите кисть снова в режим **Paint Normal** и нарисуйте глаз, нос, рот и складку между головой и туловищем (для последних элементов нужно уменьшить размер кисти). Должно получиться примерно так, как на рисунке 7.

Теперь остается последний штрих: надо добавить **контур** к некоторым элементам.

Для этого можно использовать инструмент  **Ink Bottle** — он позволяет не только менять цвет существующего контура, но и **обводить контуром заливку**.

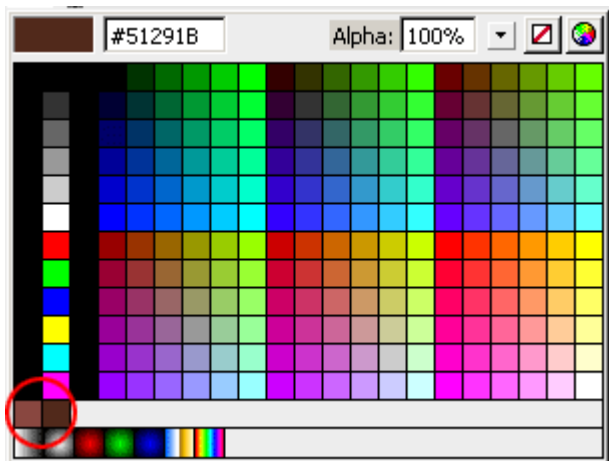
Практика : Включите инструмент , установите на панели **Properties** черный цвет и толщину линии 2. Щелкните мышью на туловище, ушах, лапах и т.д. Для того, чтобы разделить (слившиеся) ноги, используйте карандаш. Окончательный результат показан на рисунке 8.


Анимация

Теперь сделаем покадровую анимацию так, чтобы Винни-Пух моргнул глазом.

Практика : Добавьте новые ключевые кадры в кадры 2 и 3 (в них должно скопироваться изображение Винни-Пуха из кадра 1). Перейдите на кадр 2, включите инструмент кисть в режиме **Paint Fills** (закрашивать только заливки).

Вспомните, что мы добавили в палитру два новых цвета, которыми рисовали Винни-Пуха. Теперь они расположены ниже основных цветов (см. рисунок).



Практика : Выберите темно-коричневый цвет и нарисуйте Винни-Пуху веко, как на рисунке. С помощью инструмента  **Pencil** добавьте контур по нижней границе века



(рисунок).

Практика : Установите на панели **Properties** под сценой частоту кадров 2 и просмотрите фильм.

Вопрос для самостоятельной работы.

1. В чем суть покадровой анимации.
2. Что разрешает делать инструмент **Free Transform**.
3. Когда применяется **Onion Skin** («луковая кожура»).
4. Зачем нужен **Edit Multiple Frame**.
5. Что означает режим **Paint Behind** для инструмента **КИСТЬ**.

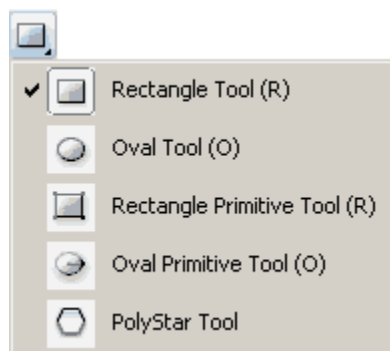
Термины: параметр Alpha, виды кадров, направляющие, режим Onion Skin, инструменте Brush и режимы его работы


Литература: [1 С.62-86, 168-177]

Тема 4. Геометрические фигуры

4.1. Геометрические фигуры


Обзор



Чтобы нарисовать прямоугольник надо включить инструмент **Rectangle** (прямоугольник), щелкнув по кнопке , на панели инструментов. Теперь можно рисовать так же, как и в редакторе *Paint* — нажать левую кнопку в одном из углов прямоугольника, растянуть и отпустить.

Если при растяжении удерживать клавишу **Shift**, получается **квадрат**.

При нажатии **Alt** прямоугольник рисуется не из угла, а из центра (точки пересечения диагоналей).

Маленькая черная стрелка в правом нижнем углу говорит о том, что одна кнопка позволяет включать несколько инструментов. Если немного задержать мышку на кнопке , при нажатой левой кнопке, мы увидим дополнительное меню, из которого сможем выбрать нужный вариант:

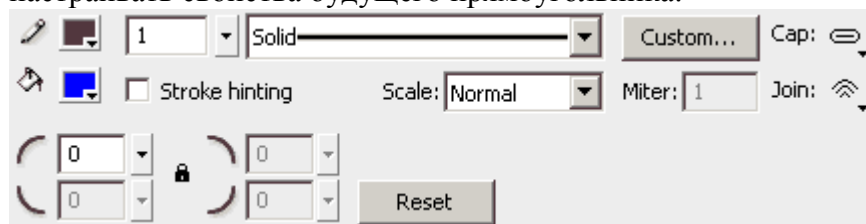
- **Rectangle** (прямоугольник);
- **Oval** (овал);
- **Rectangle Primitive** (прямоугольник с настройкой);
- **Oval Primitive** (овал с настройкой);
- **PolyStar** (многоугольник, звезда).

Если при растягивании овала удерживать нажатой клавишу **Shift**, получается круг. При нажатой клавише **Alt** круг рисуется из центра.


Фигуры с настройкой можно регулировать уже после рисования: изменить радиус скругления углов прямоугольника или сделать из овала кольцо с разрезом.

Настройки

Когда выбран инструмент **Rectangle**, внизу на панели **Properties** (Свойства) можно настраивать свойства будущего прямоугольника:

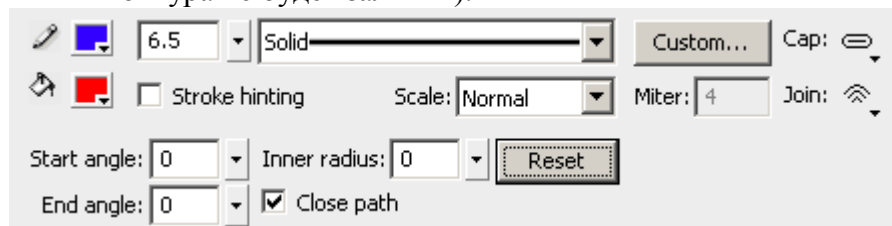


В левом верхнем углу находятся кнопки для выбора цветов контура и заливки. Окно, где на рисунке видна цифра 1, позволяет регулировать толщину линии. Список справа определяет стиль (*Solid* — сплошная линия).

С помощью элементов в левом нижнем углу можно регулировать радиус скругления углов. По умолчанию для всех углов он одинаковый, но если щелкнуть по кнопке  (отключить блокировку), можно регулировать отдельно скругление каждого из четырех углов.

На панели свойств **овала** есть некоторые новые элементы


- **Start angle** — начальный угол;
- **End angle** — конечный угол;
- **Inner radius** — радиус внутреннего выреза;
- **Close path** — флажок, показывающий, нужно ли замыкать контур (у незамкнутого контура не будет заливки).



Кнопка **Reset** сбрасывает все эти значения в нуль.

В панели дополнительных настроек геометрических фигур есть две кнопки:


 включить/отключить режим рисования объектов (**Object Drawing**);

 включить/отключить привязку к объектам (**Snap to Objects**).

Привязка к объектам означает, что стороны прямоугольника (или другой геометрической фигуры) будут «прилипать» к границам существующих объектов. Существует также «прилипание» к направляющим, сетке, пикселям изображения. Полностью настройки этой группы можно найти в меню **View—Snapping**.

Далее мы рассмотрим подробно два режима рисования геометрических фигур: стандартный режим (слияние) и режим рисования объектов.

4.2. Режим слияния

Практика : Создайте новый документ и сохраните его в папке PRACTICE\4. Включите инструмент  **Rectangle**, выберите цвет контура и цвет заливки на свой вкус. Проверьте, что режим **Object Drawing** отключен, и нарисуйте прямоугольник.

Практика : Выделите заливку и перетащите ее в сторону так, чтобы она перекрыла линию контура. Снимите выделение с заливки, а затем перетащите заливку в сторону.

Вы увидите, что часть контура, оказавшаяся под заливкой, стерта.

Практика : Отмените эти операции, нажав клавиши **Ctrl+Z** нужное число раз.

Практика : Теперь выделите контур двойным щелчком и сместите его так, чтобы он «перерезал» заливку. Перетащите мышкой одну часть заливки, затем перетащите контур так, чтобы освободить заливку. Соедините две части заливки — они «срастутся» в один объект.

Таким образом,

1. нарисованный прямоугольник состоит из контура и заливки, которые представляют собой **отдельные объекты**;
2. если заливка перекрывает контур, невидимая часть контура **стирается**;
3. если контур перекрывает заливку, он **«режет»** ее, но части заливки можно снова объединить.


Практика : Смените цвет контура, оставив тот же цвет заливки. Нарисуйте второй прямоугольник, который частично перекрывает первый. Выделите и удалите его контур. В этом случае заливки двух прямоугольников **солятся** и будут представлять собой **один объект**.


Практика : С помощью инструмента  добавьте контур к полученной фигуре.

4.3. Практикум (градиенты)

Мы нарисуем стилизованный логотип фирмы Apple:





Практика : Вставьте новый пустой ключевой кадр в кадр 2. Включите инструмент , **Rectangle**, установите прозрачный контур. С помощью панели **Color** выберите цвет заливки с параметрами **R=0, G=116, B=179** и добавьте его в палитру (пункт **Add Swatch** в меню палитры **Color**).


Практика : Проверьте, чтобы кнопка , **Object Drawing** в нижней части панели инструментов не была нажата. Установите радиус скругления углов 20 и нарисуйте квадрат в верхней части сцены (рисунок 1).

Практика : Установите белый цвет заливки, радиус скругления 0, и нарисуйте внутри белый квадрат меньшего размера, который будет изображать яблоко (рисунок 2).



Практика : С помощью инструмента  (добавление узла) добавьте новые узлы в середину каждой стороны внутреннего квадрата.

Практика : Чтобы узлы при перетаскивании не «прилипали» к сторонам объектов, отключите флажок **View—Snapping—Snap to Objects**. С помощью инструмента  **Subselection** (клавиша **A**) переместите узлы так, как показано на рисунке 3, а затем (при нажатой клавише **Alt**) сделайте их гладкими (рисунок 4).

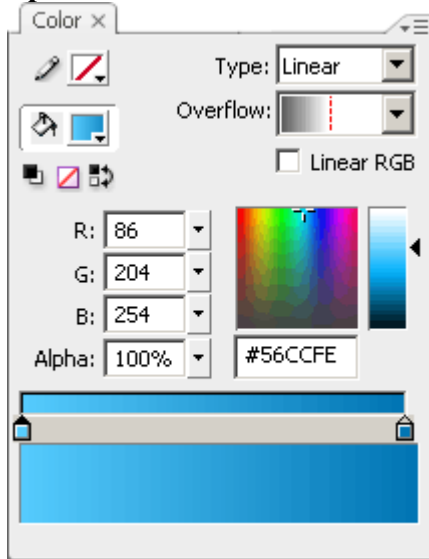
Практика : Теперь нарисуем листик над яблоком. Постройте прямоугольник белого цвета без контура и преобразуйте его в листик, перетаскивая и сглаживая узлы с помощью инструмента  **Subselection** (рисунок 5).

Практика : Выберите из палитры темно-синий цвет заливки, такой же, как у первого квадрата (когда появится палитра, можно просто щелкнуть на нем мышью). Постройте круг без контура, который «вырезает» часть яблока (рисунок 6).




Практика : Удалите две белые области (яблоко и листик). Выделите все оставшиеся (синие) области и объедините их в один объект, выбрав пункт меню **Modify—Combine Objects—Union**.


Градиенты




Градиентом называется плавный переход между двумя или несколькими цветами. Различают **линейные** градиенты (полосы) и **радиальные** (идущие от некоторого центра). Для создания градиентов используется панель **Colors**.

Сейчас включена кнопка , поэтому строится градиентная заливка (а не контур). В правом верхнем углу надо выбрать тип заливки:

- **None** — нет заливки;
- **Linear** — линейный (полосовой) градиент;
- **Radial** — радиальный градиент;
- **Bitmap** — заливка узором.



Практика : Включите инструмент  **Selection** и выделите нарисованный объект. Перейдите на панель **Color** и выберите линейный тип градиента (**Linear**).

Цвета задаются движками  в нижней части панели. Для того, чтобы изменить цвет, нужно сделать двойной щелчок по движку (откроется палитра).

Можно создать градиент из нескольких цветов. Для добавления нового движка нужно щелкнуть в нужном месте на линии градиента. Движки можно перетаскивать мышкой. Для удаления движка (цвета из градиента) надо просто стащить его вниз с линии градиента. Построенный градиент можно добавить в палитру, выбрав команду **Add Swatch** из меню панели **Color**. Градиенты отображаются в самой нижней строке палитры.

Практика : Выберите для обоих движков синий цвет, добавленный в палитру в начале упражнения. Для того, чтобы сделать цвет левого движка светлее, выделите его и переместите треугольный черный движок (у правой границы панели) вверх. Этот движок регулирует яркость цвета (параметр **Brightness** в цветовой модели HSB).




Теперь фигура залита градиентом слева направо, нам же нужно, чтобы верхняя часть была светлая, а нижняя — темнее. Для **преобразования градиента** (размер, направление, угол поворота) служит инструмент  **Gradient Transform**, который включается с помощью той же кнопки, что и  **Free Transform**. Для быстрого включения можно также использовать



клавишу **F**.

Для регулировки используются три элемента: белый кружок обозначает **центр градиента**, перетащив окружность со стрелкой можно изменить **угол поворота**, а квадрат со стрелкой позволяет менять **размер**.

Практика : Включите инструмент  **Gradient Transform** и сделайте так, чтобы градиент шел сверху вниз.

Отражение

Практика : Постройте отражение объекта. Для этого нужно скопировать его (перетащив при нажатой клавише **Alt**) и применить команду меню **Modify—Transform—Flip Vertical**. Перетащите копию точно под первый объект.


Практика : Включите инструмент  **Free Transform** и уменьшите высоту копии примерно в 2 раза.

Практика : Перейдите на панель **Color** и измените прозрачность цветов градиента для отражения: установите **Alpha=0%** для левого движка и **Alpha=50%** для правого. Вот так должен выглядеть окончательный результат:



4.4. Режим рисования объектов

Практика : Нарисуйте какую-нибудь фигуру. Скопируйте кадр 1 в кадр 3, перетащив его на временной шкале при нажатой клавише **Alt**.

Практика : Включите режим **Object Drawing** (рисование объектов), щелкнув по кнопке  в нижней части панели инструментов. Нарисуйте прямоугольник, который частично перекроет существующую фигуру.

Перетаскивая прямоугольник, мы видим, что

1. контур и заливка составляют одно целое (один **объект**) и выделяются одним щелчком;
2. объект не сливается с другими заливками и контурами, не «режет» их.





Чтобы сменить цвет и стиль контура или заливки объекта достаточно выделить его и изменить свойства на панели **Properties**.

Щелкнув дважды на таком объекте, мы попадем в **режим редактирования**, где контур и заливка — отдельные элементы, как в режиме слияния. При этом над сценой появляется надпись **Drawing Object**:



Это говорит о том, что мы редактируем нарисованный объект, находящийся на сцене *Scene 1*. Теперь можно добавлять, удалять и перекрашивать контуры и заливки. Заметим, что в объект может входить более одного контура и одной заливки. Однако нельзя добавлять другой объект внутри объекта — при этом главный объект будет преобразован в **группу**, которая имеет несколько другие свойства (например, для группы невозможна анимация формы). Для того, чтобы **разбить нарисованный объект** на отдельные составляющие, надо выделить его и выбрать команду **Break Apart** (разбить на части) из контекстного меню или из меню **Modify**.

Для возврата к сцене щелкните на тексте **Scene 1** или на голубой стрелке слева от него.

Важно : Режим рисования объектов можно включить также для ранее изученных инструментов  **Pencil**,  **Line**,  **Pen** и  **Brush**. При этом создаются объекты, которые «никому не мешают».

Нарисованный объект можно построить и **после** того, как фигура была нарисована. Можно также объединить в один объект контуры и заливки, нарисованные независимо. Для этого нужно выделить все необходимые контуры и заливки и выбрать команду **Modify—Combine Objects—Union**. Именно это мы сделали чуть раньше с логотипом *Apple*, когда строили его отражение.

Изменение порядка объектов

Для того, чтобы преобразовать фигуру или несколько фигур в объект рисования, надо выделить их и выбрать пункт меню **Modify—Combine Objects—Union**.

Практика : Преобразуйте в объект контур и заливку фигуры, полученной ранее слиянием.

Для выделения используйте двойной щелчок по заливке при включенном инструменте .

Если теперь перетащить новый объект на объект-прямоугольник, вы увидите, что новый объект оказался сверху. Если нужно изменить порядок объектов, используют меню **Modify—Arrange** или группу команд **Arrange** из контекстного меню:

- **Bring to Front** — на самый верх;
- **Bring Forward** — на одну позицию вверх;
- **Send Backward** — на одну позицию вниз;
- **Send to Back** — в самый низ.


Эти операции применяются к выделенным объектам.

Практика : Сделайте так, чтобы прямоугольник оказался выше второй фигуры.

4.5. Фигуры с настройкой

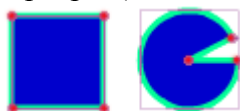
Фигуры с настройкой (так называемые «примитивные фигуры») — это новшество версии *Flash CS3*. Существуют две таких фигуры: Они включаются с помощью той же кнопки, что и прямоугольник.


 **Rectangle Primitive** (прямоугольник с настройкой)

 **Oval Primitive** (овал с настройкой)

Их отличие от обычных прямоугольника и овала состоит в том, что все параметры (в том числе радиусы скругления углов прямоугольника, начальный и конечный углы и внутренний радиус овала) регулируются **после** того, как они построены.

Если выделить такую фигуру инструментом  **Selection**, на контуре появляются точечные маркеры (они мигают на рисунке ниже).



Потянув за угловые маркеры прямоугольника, можно изменить **скругление углов**. Если нужно независимо настраивать скругление каждого угла, снимите блокировку, щелкнув по кнопке  на панели **Properties**.


Маркеры на контуре овала позволяют изменить **начальный и конечный углы** сектора, а маркер в центре овала служит для изменения **внутреннего радиуса** (можно сделать кольцо). Все свойства выделенного объекта с настройкой можно изменять также и с помощью панели **Properties**.


Чтобы преобразовать такой объект в пару «контур—заливка», нужно выбрать команду **Break Apart** из контекстного меню или из меню **Modify**. После этого «настраиваемость» фигур теряется.

4.6. Практикум (лассо, фильтры)

Мы нарисуем логотип **Adobe CS3**, зальем его градиентом с двумя переходами цветов и придадим фигуре объемный вид с помощью фильтров:



Практика : Создайте пустой ключевой кадр в кадре 4. Включите инструмент  **Rectangle Primitive** и режим **Object Drawing**. Задайте прозрачный цвет контура и темно-красный цвет заливки. Нарисуйте квадрат (удерживая клавишу **Shift**).

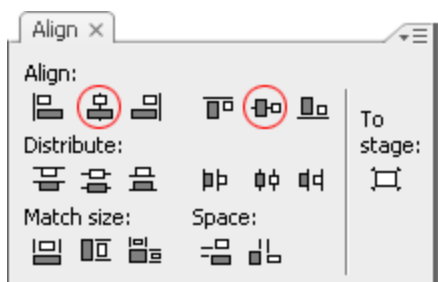
Теперь надо отрегулировать кривизну углов с помощью инструмента  **Selection**. Чтобы временно включить этот инструмент, можно нажать и удерживать клавишу **Ctrl**. Если отпустить **Ctrl**, снова включится инструмент, который был активен до этого.

Практика : Удерживая клавишу **Ctrl**, перетащите узловые точки на углах так, чтобы радиус скругления был около 20.

Практика : Снимите выделение с квадрата, щелкнув на свободном месте при нажатой клавише **Ctrl**. Установите белый цвет заливки и нарисуйте еще один квадрат меньшего размера внутри первого. Установите для него радиус скругления углов около 10.

Панель Align

Теперь нужно выровнять квадраты так, чтобы их центры совпадали. Для этого используем панель выравнивания, которая выводится на экран при выборе пункта меню **Window—Align**



На этой панели четыре группы кнопок:

- **Align** — **выравнивание** выделенных объектов (по левой, правой, верхней или нижней границе, по центру);
- **Distribute** — распределение **на равных расстояниях** (по какой-либо границе или по середине);
- **Match size** — увеличение **размера** выделенных объектов до размера большего из них;
- **Space** — выравнивание расстояний **между** объектами.

Включив кнопку **To Stage** можно выравнивать объекты относительно сцены (а не относительно друг друга).

Практика : Выделите оба квадрата (щелкнув на них при нажатой клавише **Shift** или обводя в рамку инструментом **Selection**). Щелкнув на выделенных кнопках (см. рисунок выше), выровняйте их по центру вертикально и горизонтально.

Слияние фигур

Практика : Снимите выделение с прямоугольников. Включите инструмент **Rectangle** (без регулировки), выберите красный цвет (такой же, как у нижнего квадрата) и установите скругление углов 0. Нарисуйте квадрат, который точно закроет белый квадрат.

Практика : Включите инструмент **Subselection**, выделите новый квадрат и перетащите две верхние узловые точки к центру, чтобы образовался контур буквы **A**.

Вот что у вас должно получиться:



Практика : Выделите все фигуры (обведите их рамкой при нажатой клавише **Ctrl**) и объедините их в одну, выбрав команду **Modify—Combine Objects—Union**.


Практика : Разбейте фигуру на составляющие, выбрав команду **Break Apart** из контекстного меню.




Теперь, щелкая мышкой в различных областях фигуры, вы увидите, что она состоит из трех заливок: темно-красной основной части и двух белых треугольников.

Практика : Удалите белые треугольники (выделите их и нажмите клавишу **Delete**).

Инструмент Lasso


Теперь нужно вырезать некоторую область внутри буквы **A**. Для этого мы выделим ее и удалим, нажав клавишу **Delete**.


Для выделения областей используют инструмент  **Lasso**, которым надо «обвести» нужную область. При этом в нижней части панели инструментов появляются три кнопки:

-  «волшебная палочка» (используется при работе с растровыми рисунками);
-  параметры «волшебной палочки»;
-  **Polygon Mode**, режим многоугольника.


Для нас будет интересен именно режим многоугольника, который позволит выделить и удалить внутреннюю область:



Практика : Включите инструмент  **Lasso** в режиме многоугольника. Выделите нужную область внутри буквы А, щелкая последовательно на ее углах. Двойным щелчком завершите выделение. нажав клавишу **Delete**, удалите выделенную область.

Если вы поставили узловые точки немного не там, где нужно, их можно переместить инструментом  **Subselection**.

Теперь зальем фигуру градиентом так, чтобы левый верхний и правый нижний углы были немного поярче.

Практика : Выделите фигуру, на панели **Color** постройте линейный градиент **Linear** с тремя цветами: в центре темно-красный цвет **R=135, G=8, B=30**, а по краям — цвет того же оттенка и насыщенности, но меньшей яркости: **R=243, G=63, B=100**. С помощью инструмента  **Gradient Transform** (клавиша **F**) разверните градиент по диагонали — из левого верхнего угла в правый нижний.

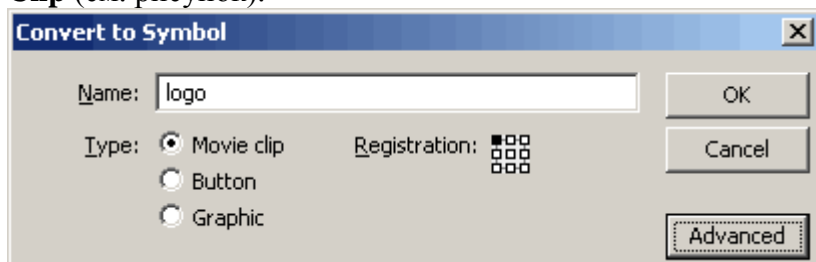
Вот результат этого этапа:




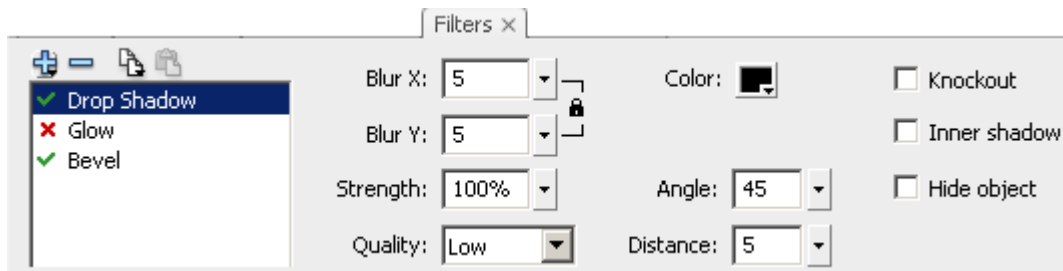
Фильтры

Чтобы добавить объемность фигуре, можно использовать специальные преобразования, которые называются **фильтры**. Фильтры можно применить только к тексту, кнопке или клипу (вспомогательному фильму, который является частью основного фильма). Разговор о них еще впереди, сейчас мы просто преобразуем заливку в клип для того, чтобы добавить объемность.

Практика : Выделите фигуру и нажмите кнопку **F8** (или выберите пункт меню **Modify—Convert to Symbol**). В появившемся окне введите имя **logo** и выберите тип символа **Movie Clip** (см. рисунок).



Практика : В нижней части окна программы (под сценой) выберите вкладку **Filters** (фильтры) и добавьте с помощью кнопки  фильтры **Bevel** (рельеф), **Glow** (свечение) и **Drop Shadow** (падающая тень) со стандартными настройками



В правой части панели **Filters** можно менять настройки выделенного фильтра, а кнопка удаляет фильтр. Значок **×** слева от названия фильтра говорит о том, что фильтр включен. Если по нему щелкнуть, фильтр временно отключается и появляется значок **+**. Повторный щелчок в этом же месте снова включит фильтр.

Отметим, что применение фильтров — обратимая операция, изображение при этом не портится.

Вот окончательный результат работы:



Повторение темы 4 и контрольная работа - 2 часа.

Контрольная работа

Практика : Создайте пустой ключевой кадр в кадре 5. Постройте объемный логотип фирмы *Microsoft*. В качестве подсказки используйте анимированный рисунок, показанный ниже.



Вопрос для самостоятельной работы.

1. Какой инструмент применяется для рисования овала.
2. Что такое градиент.
3. Какие виды градиентов можно выбрать в Adobe Flash CS3.
4. Что разрешает делать режим рисования фигуры с настраиванием.
5. Для чего употребится панель **Align**.

Термины: Привязка к объектам, градиент, фильтр, виды анимации, форма в анимации.

Литература: [[1](#) С.62-86, 168-177]

Тема 5. Анимация формы

5.1. Введение

Раньше мы использовали покадровую анимацию, когда каждый кадр фильма приходится рисовать вручную. Если фильм большой, это требует очень много времени. Чтобы автоматизировать процесс, в программе *Flash* можно задать только рисунки в ключевых кадрах, а промежуточные изображения программа построит сама.



Существует два вида автоматической анимации: **анимация формы и анимация движения**.



В ходе этого урока мы познакомимся подробно с **анимацией формы**.

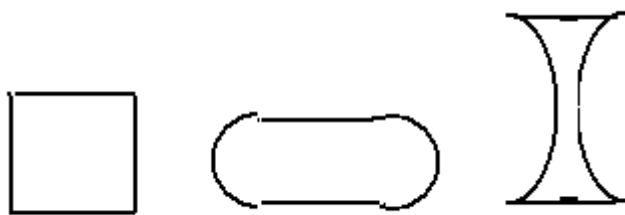
Форма — это контур, заливка или несколько контуров и заливок. Анимация формы позволяет

- плавно преобразовать одну **форму** в другую (*морфинг*);
- изменить **цвет** формы;
- **переместить** форму.

Практика: Создайте новый документ, установите размеры поля 400 на 400 пикселей и сохраните его в папке PRACTICE\5 под именем contour fla.

Практика: Выберите инструмент  (прямоугольник), установите черный цвет контура, без заливки. Обязательно **отключите режим рисования объектов** (кнопка  в нижней части панели инструментов не должна быть нажата). Нарисуйте в центре поля квадрат размером примерно 300 на 300.

Практика : Вставьте новые ключевые кадры в кадры 10, 20 и 30 (клавиша **F6**). Используя инструмент  **Selection** и  **Free transform**, измените контуры в кадрах 10 и 20 так, как показано на рисунках ниже. Кадр 30 оставьте без изменений (он совпадает с кадром 1).



Кадр 1,30

кадр 10

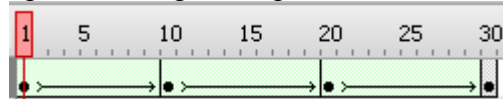
кадр 20

Теперь мы создадим плавные переходы между этими формами, причем рисовать вручную ничего не придется.

Практика : Чтобы включить анимацию формы для первого перехода, щелкните правой кнопкой мыши по кадру 1 и выберите из контекстного меню пункт **Create Shape Tween** (создать анимацию формы). Повторите то же самое для кадров 10 и 20.

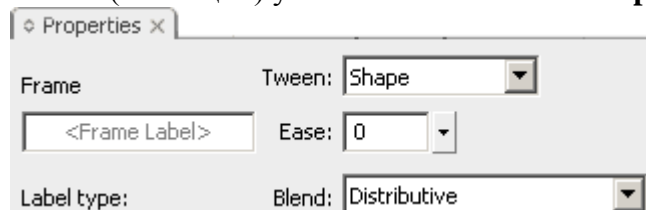
Слово *tween* — это жаргонное сокращение от английского *in between* (между). Это выражение пришло из классической анимации, где ведущие художники рисовали только наиболее важные (ключевые) сцены мультфильма, а их менее талантливые коллеги дорисовывали промежуточные кадры.

Если вы сделали все правильно, промежуточные кадры имеют зеленоватый цвет и через них проходит черная стрелка от начального кадра к конечному:



Практика : Проверьте ролик, нажав клавиши **Ctrl+Enter**.

Если выделить любой из кадров и посмотреть на панель **Properties**, вы увидите, что в списке **Tween** (анимации) установлено значение **Shape** (*shape* — форма).



К анимации относятся также параметры **Ease** и **Blend**. Обычно анимация выполняется с постоянной скоростью. Если надо постепенно замедлять анимацию, используют положительные значения параметра **Ease** (от 1 до 100), для ускорения — отрицательные (от -1 до -100).

Параметр **Blend** имеет два возможных значения: **Distributive** (плавные контуры) и **Angular** (сохранение углов).

5.2. Контрольные точки

Практика : Создайте новый документ, установите размер 400 на 400 пикселей. Сохраните его в папке PRACTICE\5 под именем shapes fla.

Практика : Вытащите на экран линейки, выбрав пункт меню **View—Rulers**. Установите две горизонтальных и две вертикальных направляющих, так чтобы они ограничили квадрат размером примерно 300 на 300.

Практика : Нарисуйте синий круг без границы, вписав его в квадрат.

Практика : Добавьте пустой ключевой кадр в кадр 10 (клавиша **F7**). Нарисуйте красный квадрат без контура, который выровнен по направляющим. Добавьте анимацию формы к кадру 1. Просмотрите результат.

Когда круг переходит в квадрат, видно некоторое вращение. Чтобы сделать анимацию более красивой, мы поможем программе, установив **контрольные точки** (*hints*).

Контрольные точки создаются на первом кадре анимации и устанавливаются мышью в характерные точки формы. Если после этого перейти на последний кадр интервала анимации, мы увидим те же точки, которые можно перетащить мышкой в нужные места.

Контрольные точки обозначаются латинскими буквами (*a, b, c, ...*). Точка *a* на первом кадре «приходит» туда, куда установлена точка *a* на последнем кадре.

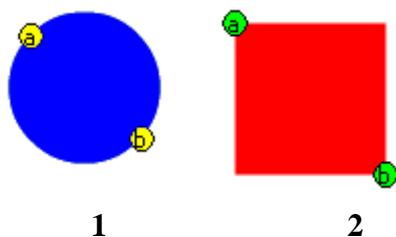
Контрольные точки бывают трех цветов:

- **красный** — контрольная точка создана, но еще не перемещалась;
- **желтый** — контрольная точка на первом кадре анимации, установленная на контуре фигуры;
- **зеленый** — установленная контрольная точка на последнем кадре анимации.

В программе предусмотрены следующие операции с контрольными точками:

- **добавить** точку — нажать клавиши **Ctrl+Shift+H** или выбрать пункт меню **Modify—Shape—Add Shape Hint**;
- **удалить** точку — щелкнуть правой кнопкой мыши на точке и выбрать пункт **Remove Hint** из контекстного меню; удалить все точки можно с помощью пункта меню **Modify—Shape—Remove All Hints**;
- **показать** (скрыть) контрольные точки можно с помощью пункта **View—Show Shape Hints** главного меню.

Практика : Перейдите на кадр 1 и установите две контрольные точки так, как показано на рисунке 1. Затем перейдите на кадр 10 и переместите контрольные точки в противоположные углы квадрата (рисунок 2).



Новый урок – создаем анимацию как на предыдущем уроке

5.3. Оптимизация контура

Практика : Добавьте пустой ключевой кадр в кадр 20 и нарисуйте кистью в пределах квадрата из направляющих контур бабочки синего цвета (рисунок 3 ниже).


Если контур сложный (содержит много узлов), компьютер тратит много ресурсов, чтобы рассчитать все промежуточные точки и нарисовать его на экране. Поэтому для ускорения работы ролика и снижения нагрузки на процессор желательно оптимизировать контур, **удалив узлы**, которые мало влияют на форму.



3

Практика : Выделив бабочку, выберите команду **Modify—Shape—Optimize** из верхнего меню. В появившемся окне установите движок **Smoothing** (сглаживание) в положение **Maximum**. После применения команды будет выдано сообщение о том, сколько узлов было и сколько осталось после оптимизации.

Скорее всего, бабочка получится неровная. Чтобы сделать ее симметричной, мыотрежем правую половину и скопируем на ее место отраженную левую половину.

Практика : С помощью инструмента  **Lasso** в режиме многоугольника выделите правую половину бабочки (линия разреза должна проходить посередине тельца) и удалите ее. Затем скопируйте левую половину на свободное место (перетащив при нажатой клавише **Alt**) и отразите от вертикальной оси (**Modify—Transform—Flip Horizontal**). Соедините две половинки и залейте образовавшуюся единую фигуру градиентом *синий—черный—синий* (рисунок 5).



5

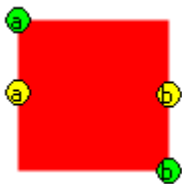
Практика : Добавьте к кадру 10 анимацию формы (переход квадрата в бабочку).

Просмотрите ролик.

Вы заметите, что анимация на втором интервале получилась не очень красивая. Чтобы ее улучшить, мы добавим контрольные точки.

Практика : Перейдите на кадр 10. Вы должны увидеть две зеленые контрольные точки в углах квадрата (от предыдущей анимации). Если вы их не видите, выберите пункт меню **View—Show Shape Hints**. Добавьте две контрольных точки и перетащите их на середины сторон квадрата (точки желтого цвета на рисунке 6).

Практика : Перейдите на кадр 20 и перетащите контрольные точки на контур бабочки, как показано на рисунке 7. Просмотрите ролик — анимация должна улучшиться.



6



7

Важно : Анимация формы — сложный процесс, который иногда дает сбой. Например, при установке нескольких контрольных точек изображение в промежуточных кадрах может вообще исчезнуть. Поэтому рекомендуется использовать наименьшее количество точек (лучше две) и расставлять их в порядке обхода контура против часовой стрелки.

5.4. Цвет и движение

Теперь мы проверим еще два варианта анимации формы: **изменение цвета** и **движение**.

Практика : Вставьте новый ключевой кадр в кадр 30, скопировав в него содержимое кадра 20 (клавиша **F6**). Выделите бабочку и с помощью панели **Color** поменяйте цвета градиента: установите переходы *красный—желтый—красный*.

Практика : Вставьте новый ключевой кадр в кадр 40, скопировав в него содержимое кадра 30 (клавиша **F6**). Перетащите бабочку вертикально вверх так, чтобы она оказалась за пределами сцены. Включите анимацию формы для кадров 20 и 30. Для того, чтобы бабочка улетала с возрастающей скоростью, в панели **Properties** установите параметр **Ease** равным **-100**.

Практика : Вставьте новый промежуточный кадр в кадр 45 (клавиша **F5**), чтобы сделать паузу перед новым циклом анимации.



Практика : Теперь перейдите в кадр 40 и попробуйте повернуть бабочку на некоторый угол. Просмотрев ролик, вы увидите, что при анимации форма бабочки искажается, фактически анимация получается неудачной. Чтобы исправить ситуацию, в этом случае надо использовать анимацию движения, о которой рассказывается в следующей теме.

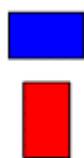
Создаем анимацию – Бабочка машет крыльями.

Практика: Создаем новый документ. Рисуем бабочку. В кадре 5 и 10 делаем ключевые кадры. В кадре 5 уменьшаем по горизонтали форму бабочки. В кадре 10 бабочка такая же как и на кадре 1. Включаем анимацию формы.

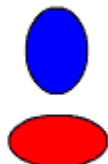
5.5 Слои

Давайте попробуем сделать одновременную анимацию нескольких объектов.

Практика : Создайте новый документ. Включите инструмент  (прямоугольник) в режиме рисования объектов (включите кнопку  в нижней части панели инструментов). Нарисуйте два прямоугольника с черным контуром и разными цветами заливки, как показано на рисунке 1.



1



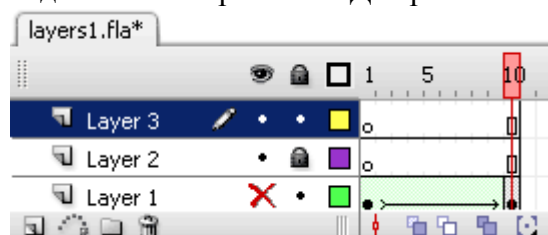
2

Практика : Вставьте пустой ключевой кадр в кадр 10 и нарисуйте два овала, как показано на рисунке 2. Включите анимацию формы для кадра 1.



Можно было бы предполагать, что программа «сообразит», что синий прямоугольник надо преобразовать в синий овал, а красный прямоугольник — соответственно в красный овал. Но вместо этого в середине интервала будет мешанина из четырех объектов.

Чтобы решить возникшую проблему, мы явно укажем, что здесь два независимых объекта, для каждого из которых надо строить переход отдельно. Для этого придется каждую анимацию расположить на **отдельном слое**.



В программе *Flash*, как и во многих других программах для работы с графикой, документ состоит из нескольких независимых слоев, на каждом из которых может быть отдельное изображение. Для работы со слоями служит палитра слева от временной шкалы:




В этом документе три слоя с именами *Layer 1*, *Layer 2* и *Layer 3*. Для того, чтобы **сменить имя**, надо сделать двойной щелчок на имени слоя. Здесь самый нижний слой — это слой *Layer 1*, выше него расположен *Layer 2* и на самом верху — слой *Layer 3*. **Порядок слоев** можно менять, перетаскивая их мышкой. Активный слой, на котором идет рисование, выделен синим цветом и отмечен значком карандаша. Значки над палитрой имеют следующее значение:

-  **видимость** слоев;
-  **блокировка** слоев (защита от изменений);
- контурное изображение** (показываются только контуры объектов).

Щелчок по одному из этих значков позволяет установить (снять) эти режимы сразу у всех слоев. Если же надо применить режим к одному слою, следует щелкнуть на точке ниже соответствующего значка. На рисунке выше активным является слой *Layer 3*, слой *Layer 2* заблокирован, а слой *Layer 1* невидим.

Чтобы создать **новый слой** выше существующего, нужно щелкнуть по кнопке  в нижней строке палитры. Кнопка  позволяет создать слой специального типа — **направляющую** для анимации движения.

Слои можно организовывать в **папки**. Это очень важно в больших проектах, где бывает несколько десятков слоев. Чтобы **создать папку**, нужно щелкнуть по кнопке . Затем перетащите нужные слои на созданную папку.

Для удаления выделенного слоя (или папки) надо щелкнуть по кнопке .

Практика : Переименуйте слой *Layer 1*, пусть он называется *Синий*. Создайте новый слой с именем *Красный*.

Практика : Перейдите к кадру 1. Выделите красный прямоугольник и удалите его в буфер обмена (**Ctrl+X**). Затем перейдите в кадр 1 слоя *Красный* и вставьте фигуру из буфера обмена, нажав комбинацию клавиш **Ctrl+Shift+V** (меню **Edit—Paste in Place**).

В отличие от стандартной комбинации **Ctrl+V**, этот прием вставляет объект не в центр поля, а в то же место, где он был (возможно, на другой слой).

Практика : Отключите видимость слоя *Синий*, чтобы убедиться, что теперь красный прямоугольник находится на слое *Красный*.

Практика : Вставьте в кадре 10 новый пустой ключевой кадр. Таким же образом переместите в него красный овал из кадра 10 слоя *Синий*.

Практика : Включите анимацию формы для кадра 1 слоя *Красный*. Проверьте ролик. Теперь результат стал значительно лучше.

Вообще при создании анимационных роликов в программе *Flash* рекомендуется **размещать каждый объект на отдельном слое**.

Анимация движения .(основы)

Практика: создаем новый документ. В кадре 1 в левой части сцена рисуем прямоугольник с линейным градиентом. Вставляем ключевые кадры в кадр 10 и 20. В кадре 10 передвинуть прямоугольник к противоположной границе сцены. Чтобы случайно не сдвинуть его влево или вправо, лучше использовать для перемещения стрелку «вправо». При нажатой клавише **Shift** скорость перемещения увеличивается.



Кадр 1 и 20

Кадр 10

Теперь программа сможет автоматически построить все промежуточные кадры анимации между кадрами 1—10 и 10—20. Это **анимация движения**, поскольку форма объекта не меняется.

Практика: Щелкните правой кнопкой мыши по кадру 1 и выберите пункт меню **Create Motion Tween** (создать анимацию движения). Сделайте то же самое для кадра 10.

Просмотрите ролик.

Более подробно анимацию движения рассмотрим позже.

5.6 Звук (проработать дополнительно дома)

Присутствие звука часто позволяет усилить действие фильма, поскольку кроме зрительного используется еще один канал передачи информации — звуковой.

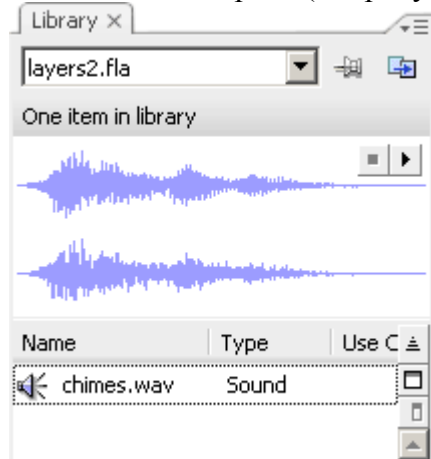
Программа позволяет добавлять звуки четырех широко распространенных форматов:

- **WAV** (*WAVEform audio format*) — стандартный формат *Windows*;
- **MP3** (*MPEG-1 Audio Layer 3*) — формат, использующий сжатие за счет удаления составляющих звука, которые человек не слышит;
- **AIFF** (*Audio Interchange File Format*) — звуковой формат фирмы *Apple*;
- **AU** (*AUdio file format*) — формат кодирования звука, разработанный фирмой *Sun*.

Перед тем, как вставить звук из файла, его нужно загрузить (*импортировать*) в **библиотеку**. Библиотека — это набор элементов (рисунков, звуков, символов и т.п.), которые используются в ролике.

Практика : Выберите команду меню **File—Import—Import to Library**, найдите файл *chimes.wav* и загрузите его в библиотеку.

Теперь звук находится в библиотеке файла и его можно увидеть на панели **Library** в правой нижней части экрана (см. рисунок).




В библиотеке сейчас один элемент типа **Sound** (звук). Он выделен, и в средней части панели мы видим диаграммы левого и правого каналов (стереозвук).

Щелкнув по кнопке  можно **прослушать** запись.

Щелкнув правой кнопкой мыши на звуке в библиотеке и выбрав команду **Properties**, можно просмотреть его свойства и изменить степень сжатия (для уменьшения объема файла).

В принципе можно связать звук с любым слоем фильма. Однако обычно удобно использовать для этого отдельный слой, чтобы не путать звуки с другим содержимым.

Практика : Создайте новый слой и назовите его **Звук**. Выделите кадр 1 этого слоя и перетащите на сцену значок звука  из библиотеки.

На временной шкале на кадрах этого слоя будет изображена временная диаграмма, отражающая изменение звука во времени:

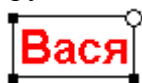


Практика : Проиграйте ролик, включив наушники или колонки.

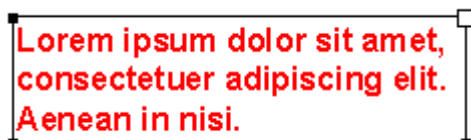
Подробнее использование звука мы изучим на следующих уроках.

5.7. Текст


Для ввода текста используется инструмент **T Text**. Чтобы ввести **однострочный текст**, нужно включить инструмент **T**, щелкнуть в нужном месте и набрать текст на клавиатуре. Вы увидите, что при этом размер текстового поля будет автоматически изменяться. Белый кружок в правом верхнем углу рамки говорит о том, что это однострочный текст.



Если растянуть рамку за угол, однострочный текст превратится в **текстовый блок**, ширина которого не изменяется. Когда очередная строка заполняется, происходит автоматический переход на следующую. В правом верхнем углу рамки появляется белый квадрат. Если щелкнуть по нему дважды, блок превращается в однострочный текст.

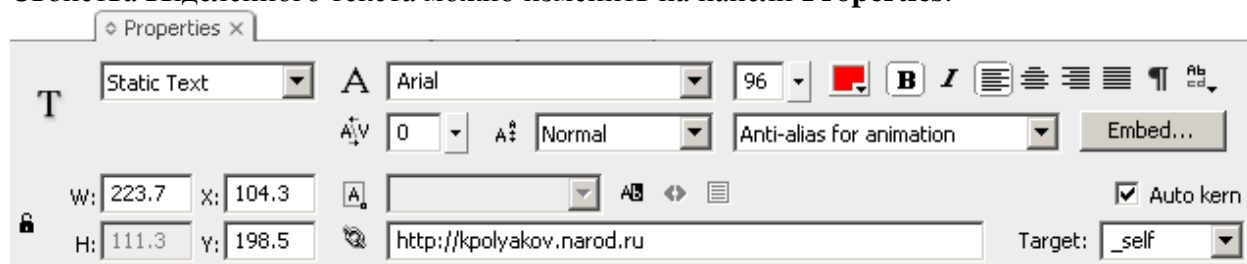


Можно и сразу добавить текстовый блок: включив инструмент **T**, обвести мышкой рамку нужной ширины.

Изменять размеры блока и перемещать его можно с помощью инструмента . Напомним, что он временно включается, если нажать клавишу **Ctrl** при любом другом включенном инструменте.

Свойства текста

Свойства выделенного текста можно изменить на панели **Properties**:






Выпадающий список в левом верхнем углу позволяет выбрать **тип текста**:


- **Static Text** — неизменяемая надпись;
- **Dynamic Text** — текст, который можно менять во время работы ролика из программы (обычно используется для информационных сообщений);
- **Input Text** — поле для ввода текста.

В этом уроке мы будем использовать только текстовые надписи (**Static Text**).

В верхней строке панели находятся стандартные элементы для выбора шрифта, размера (в пунктах), стиля, выравнивания. Если неизвестно, какие шрифты установлены у пользователя, желательно выбирать системные шрифты **_sans** (рубленный шрифт, без засечек, для заголовков и мелкого текста), **_serif** (шрифт с засечками, для облегчения чтения длинного текста) или **_typewriter** (имитация пишущей машинки, все буквы одинаковой ширины).

Кнопка  позволяет настроить свойства абзаца для текстового блока (отступы, красную строку, интервал между строчками). С помощью кнопки  можно изменить направление текста (сверху вниз).


В окошке справа от знака  регулируется **интервал между символами** (можно сделать более плотный или более разреженный текст).

Список, обозначенный символом , содержит варианты **Normal** (обычный текст), **Subscript** (нижний индекс) и **Superscript** (верхний индекс).

Еще правее можно выбрать тип коррекции мелкого текста (*anti-aliasing*). Сейчас выбран вариант **Anti-alias for animation** — сглаживание для анимации. Если нужно улучшить читаемость мелких букв (без анимации), выбирают **Anti-alias for readability**.

Кнопка  разрешает выделение текста мышью.

Флажок **Auto kern** включает автоматическую коррекцию расстояния между особыми парами символов. Эффект можно увидеть, если набрать буквы **A** и **V** рядом и сильно увеличить размер шрифта.

Если ввести адрес Web-страницы в поле справа от знака , текст становится гиперссылкой. Список **Target** определяет, где открывается эта страница:

- **_self** — в том же окне браузера;
- **_blank** — в новом окне;
- **_parent** — в родительском окне (для фреймов);
- **_top** — в главном окне (для фреймов).

Если используется любой шрифт, кроме системных, начертания букв можно сохранить внутри *Flash*-ролика (**внедрить** шрифт в документ). Это гарантирует, что символы будут отображаться точно так, как задумал автор, даже если на компьютере у пользователя нет этого шрифта. Однако при этом несколько увеличивается объем файла.

Для внедрения шрифтов служит кнопка **Embed**, которая позволяет сохранить не все символы, а только нужные для данного текста, например, только цифры.

Для текста можно использовать фильтры (панель **Filters**).


Анимация текста

Анимация формы, которую мы изучали выше, применима и для текста. Морфинг (преобразование одной буквы в другую) сделать можно, но обычно это не имеет смысла и выглядит некрасиво. Чаще всего используют изменение размера и цвета букв. Правда, для этого приходится преобразовывать текст в формы (заливки), потеряв возможность редактировать его с помощью инструмента **T**.

Далее мы выполним анимацию слова УРА на синем фоне.

Практика : Создайте новый документ размером 500 на 200 пикселей и сохраните его.

Нарисуйте прямоугольник без рамки, закрывающий всю сцену и залейте его градиентом (снизу — черный, сверху — темно-синий). Переименуйте слой *Layer 1* в *Фон* и заблокируйте его.

Практика : Вставьте новый слой *Буквы* выше фона. С помощью инструмента **T** введите на этом слое в центре сцены текст **УРА** белого цвета размером 120 (этот размер нельзя установить с помощью движка, но можно ввести с клавиатуры). Выберите интервал между буквами 60 (элемент  на панели **Properties**) и расположите текст по центру.

Чтобы использовать анимацию формы, текст надо превратить в форму, то есть в заливку.

Для этого дважды применим команду **Break Apart** из контекстного меню или из меню **Modify** (клавиши **Ctrl+B**). В первый раз текст будет разделен на отдельные буквы, которые пока остаются текстом (их можно редактировать с помощью инструмента **T**). Во второй раз буквы преобразуются в заливки.

Практика : Превратите текст в заливки. Вставьте новые ключевые кадры в кадрах 10, 20, 30 и 40 слоя *Буквы*, и новый промежуточный кадр в кадре 40 слоя *Фон*.

Практика : Перейдите в кадр 1 и сделайте (с помощью панели **Color**) все буквы полностью прозрачными (**Alpha=0**).

Практика : Перейдите в кадр 10. Увеличьте букву *У* ровно в 2 раза (меню **Modify—Transform—Scale and Rotate—200%**), а буквы *Р* и *А* сделайте прозрачными. Аналогично в кадрах 20 и 30 сделайте увеличенными буквы *Р* и *А*.

Практика : Скопируйте кадр 1 в кадр 40 и установите для слоя *Буквы* анимацию формы в кадрах 1, 10, 20 и 30. Просмотрите результат и закройте документ, сохранив его.

Задание - сделать анимацию текста своего ИМЕНИ на фоне с градиентной заливкой.

Важно : При более сложной анимации придется разместить каждую букву на отдельном слое. Для этого надо выделить все буквы надписи и выбрать команду **Distribute to Layers** (распределить по слоям) из контекстного меню или меню **Modify—Timeline**.

Эффект освещения

Построим еще один интересный пример анимации текста. В нем имитируется освещение надписи прожектором или фонариком.

Практика : Создайте новый документ размером 700 на 300 пикселей и сохраните его.


Переименуйте слой *Layer 1* в *Фон* и добавьте на него рисунок – например *zimnii.jpg*.
Зabloкируйте слой *Фон*.


Практика : Создайте новый слой *Текст* и расположите его сверху. В правом верхнем углу введите текст *Санкт-Петербург* (шрифт *Arial*, размер 50, цвет белый, жирный).

Превратите текст в заливку (дважды **Ctrl+B**).

Практика : На панели **Color** создайте радиальный градиент (**Radial**) из трех цветов: белый, голубой, черный. При выборе цвета для среднего движка можно с помощью пипетки взять цвет с фонового рисунка. Сохраните градиент в палитре (**Add Swatch** в меню палитры **Color**).



Практика : Включите инструмент  и выберите в палитре только что созданный градиент. Выделите все буквы (для этого достаточно снять выделение с кадра 1 слоя *Текст*, а потом снова выделить его на временной шкале) и залейте этим градиентом.

Практика : Включите инструмент  (**Gradient Transform**, клавиша **F**) и настройте градиент так, чтобы его центр находился слева от букв и буква *С* была немного освещена.

Практика : Вставьте ключевой кадр в кадр 40 слоя *Текст* и промежуточный кадр в кадр 40 слоя *Фон*. Для текста измените расположение градиента так, чтобы его центр находился справа и буква *г* была чуть-чуть освещена.

Практика : Включите анимацию формы для кадра 1 слоя *Текст* и просмотрите результат.

5.8 Слои-маски

На 2-ой урок студент приносит 6 фото и делает слайд-шоу.

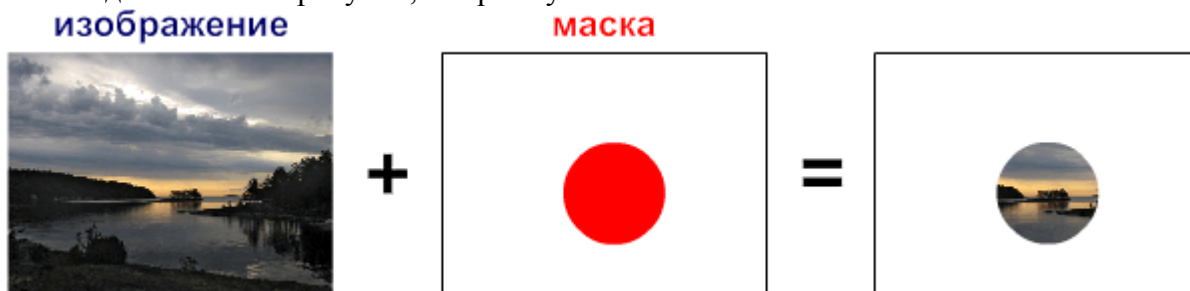
Постановка задачи


При наведении курсора мыши на рисунок начинается слайд-шоу с различными эффектами, которые мы построим с помощью анимации формы.

Для создания переходов между слайдами можно использовать два пути:


1. **обрезать** часть рисунка, которая пока не видна;
2. каким-то образом делать эту область прозрачной (невидимой).

В среде *Flash* есть возможность заблокировать некоторые части одного изображения с помощью **маски**. Маска — это объект (слой или клип), который имеет закрасненные и незакрасненные области. В отличие от карнавальной маски, через закрасненные области виден основной рисунок, а через пустые — нет:



Другими словами, заливка в маске вырезает часть рисунка, а все остальное делает прозрачным. Причем цвет заливки в маске не имеет значения, он может быть даже полупрозрачным. Более того, в **режиме слияния** (когда отключена кнопка  в нижней

части панели инструментов) на маске можно рисовать разными цветами, все области будут выделять части изображения.

Если же мы работаем в режиме **рисования объектов** (кнопка  нажата), маской становится только самый нижний объект.

Итак, для выполнения перехода мы будем использовать слой-маску, на котором с помощью анимации формы меняется закрашенная часть (заливка).

Переход «изнутри»

Практика : Откройте файл PRACTICE\5\show.fla.

На слоях *Layer 1*, *Layer 2* и *Layer 3* расположены три фотографии, которые должны сменять друг друга (проверьте это, отключив видимость верхних слоев). Нам нужно организовать три перехода, каждый из них будет занимать 10 кадров. Кроме того, еще 10 кадров каждая фотография удерживается на экране после окончания перехода. Таким образом, всего потребуется 60 кадров (3 раза по 10+10).

Первая фотография (в слое *Layer 1*) будет постоянно видима. Когда нужно, мы будем перекрывать ее другими.

Практика : В слое *Layer 1* вставьте новый промежуточный кадр (**F5**) в кадр 60.

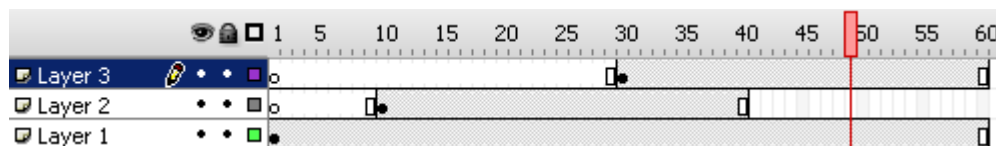
Второй снимок начинает появляться с 10-ого кадра и будет не нужен после 40-ого.

Практика : В слое *Layer 2* перетащите первый ключевой кадр в кадр 10 и вставьте новый промежуточный кадр в кадр 40.


Третья фотография держится с 30-ого по 60-ый кадр.


Практика : В слое *Layer 3* перетащите первый ключевой кадр в кадр 30. При появлении вопроса *"One or more library items already exist in the document"* выберите вариант *"Don't replace existing items"*. Вставьте новый промежуточный кадр в кадр 60.

Временная шкала должна выглядеть так:




Если запустить ролик, мы увидим, что рисунки меняются мгновенно. Займемся переходами. Первый переход (от 1-ой фотографии ко 2-ой) сделаем «изнутри», то есть видимая часть второго слайда появляется постепенно из центра первого (посмотрите еще раз готовый ролик). Нужно сделать маску в виде прямоугольника, который расширяется из центра до полного размера сцены.

Практика : Выделите слой *Layer 2* и добавьте новый слой выше него, щелкнув по кнопке  под списком слоев. Переименуйте этот слой в *Маска 2* (двойной щелчок на имени). Нажмите правую кнопку мыши на новом слое и выберите пункт **Mask** из контекстного меню — это и будет слой-маска.

Заметьте, что при этом слой *Layer 2* и его слой-маска оказались заблокированными (справа от названий появились значки ). Только в этом случае маска работает в режиме редактирования. Но в слое-маске ничего нет, поэтому сейчас мы не увидим никаких изменений.

Практика : В слое *Маска 2* добавьте новый ключевой кадр в кадр 10 (клавиша **F6**).

Разблокируйте слой и отключите режим рисования объектов, если он был включен (кнопка  не должна быть утоплена). Нарисуйте прямоугольник без рамки, занимающий всю область рисунка. Цвет заливки лучше выбрать полупрозрачным (параметр **Alpha** около 50%).

Практика : Вставьте в том же слое ключевой кадр 20 (**F6**) и удалите кадры с 21 по 60 (выделить и выбрать **Remove Frames** в контекстном меню).


Практика : Перейдите в кадр 10, выделите прямоугольник и установите на панели **Properties** его ширину 1 пиксель, высоту 1 пиксель и координаты **X=275** и **Y=206** (центр экрана). Установите для этого кадра анимацию формы и проверьте клип.

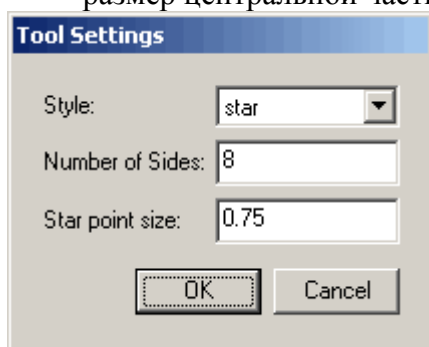
Переход «к центру»


Перейдем ко второму, наиболее сложному переходу. Здесь новый рисунок сначала появляется по краям сцены, а область старого постепенно сужается к центру. Таким образом, маска — это прямоугольник с «дыркой» в середине, причем эта «дырка» уменьшается до точки. Чтобы «дырка» была красивой, сделаем ее в форме звезды.

Практика : Добавьте в самый верх слой *Маска 3* и сделайте его маской для слоя *Layer 3*. Разблокируйте слой-маску и добавьте новый ключевой кадр 30.

Сначала «дырка» должна перекрывать всю сцену — через нее будет виден второй слайд.

Практика : Включите инструмент **PolyStar**  (многоугольник), он находится под той же кнопкой, что и прямоугольник. Щелкните на панели **Properties** по кнопке **Options** (она позволяет определить режимы) и выберите тип **star** (звезда), число вершин — 8 и размер центральной части 0.75 (75% от полного диаметра).



Практика : Включите режим рисования объектов (кнопка  в нижней части панели инструментов). Уменьшите масштаб до 25% (список в верхнем правом углу сцены), установите красный полупрозрачный цвет заливки, без контура и нарисуйте звезду, которая полностью закрывает всю сцену.

Практика : Установите полупрозрачный синий цвет заливки и нарисуйте прямоугольник, в котором целиком находится звезда. Переведите прямоугольник на задний план (**Arrange-Sent to Back** из контекстного меню).

Вот что должно получиться:



Практика : Вставьте новый ключевой кадр в кадр 40 слоя *Маска 3* (**F6**). В кадр 41 вставьте новый пустой ключевой кадр (**F7**), чтобы вообще убрать маску на последнем участке.

Практика : Перейдите в кадр 40. Выделите звезду и с помощью панели **Properties** установите высоту и ширину звезды 1 пиксель и координаты **X=275** и **Y=206** (центр экрана).

Теперь надо в обоих крайних ключевых кадрах вырезать области, которые занимает звезда. Мы использовали нарисованные объекты для того, чтобы удобнее было их расположить и настроить размеры. Теперь переведем их в простые заливки и удалим центральную часть.

Практика : Уменьшите масштаб, выделите в кадре 40 звезду и прямоугольник. Чтобы преобразовать их в простые заливки, примените команду **Break Apart** из контекстного меню. Затем увеличьте масштаб до 800%, выделите звезду (почти точка!) и удалите ее, сделав миниатюрный вырез в маске.

Практика : Перейдите в кадр 30 и таким же способом вырезайте звезду из прямоугольника (не уменьшая ее размеры!). Затем включите анимацию формы для кадра 30, заблокируйте слой *Маска 3* и просмотрите результат.

Простое перемещение

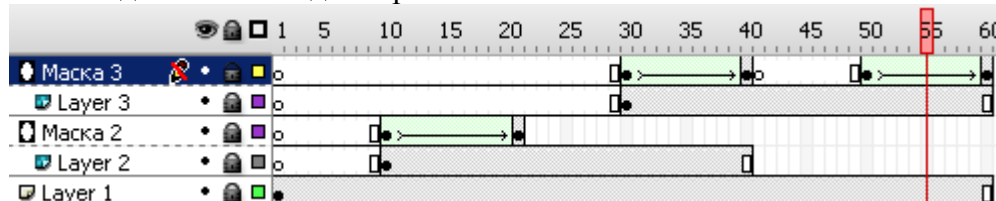
Для перехода от третьего рисунка обратно к первому мы будем также использовать слой-маску *Маска 3*. Сначала эта маска полностью открывает рисунок на слое *Layer 3*, а к 60-ому кадру полностью скрывает его, открывая таким образом слой *Layer 1* (ведь *Layer 2* вообще отключен на последнем этапе!).

Практика : Разблокируйте слой *Маска 3*, добавьте новый ключевой кадр в кадр 50.

Нарисуйте прямоугольник, разверните его на 45 градусов и измените размеры так, чтобы он перекрыл всю сцену.

Практика : Вставьте новый ключевой кадр в кадр 60 и в нем сдвиньте прямоугольник так, чтобы он оказался полностью за границами сцены. Добавьте анимацию формы для кадра 50 слоя *Маска 3* и заблокируйте этот слой.

Вот так должна выглядеть временная шкала:



Практика : Сохраните файл и просмотрите окончательный вариант.

Вопросы для самостоятельной работы.

1. Какие виды автоматической анимации применяются в Adobe Flash CS3.
2. Что разрешает делать анимация формы.
3. Зачем применяют контрольные точки на форме
4. Какие форматы звуковых файлов разрешает прибавлять Adobe Flash CS3.
5. Какие типы текста предлагает Adobe Flash CS3.
6. Для чего нужен Слой-Маска.

Термины: контрольные точки, слои, типы текста, маска.

Литература: [[1](#) С.87-99,148-159,168-189,386-399 ; [2](#) С.180-217]

4 семестр.

Тема 6. Символы. Анимация движения

6.1. Символы

Символы — это объекты, которые хранятся в библиотеке и могут использоваться в ролике много раз. Важно, что при добавлении нового символа на сцену расход памяти практически не увеличивается: описание символа хранится только один раз, в библиотеке.

У символов есть еще одно важное достоинство: при изменении символа в библиотеке сразу автоматически меняются все экземпляры.

В программе *Flash* различают три типа символов:


- **Graphic** — графическое изображение без анимации;
- **Button** — кнопка, имеющая три различных состояния (нормальное, при наведении мыши, при щелчке);
- **Movie Clip** — клип, который может включать внутреннюю анимацию.


Для всех типов символов можно применять анимацию движения, когда объект меняет положение или вращается без изменения формы.

Практика: Создайте новый документ с размером сцены 550 на 400 пикселей. Сохраните его в папке PRACTICE\6 под именем ball fla.

Практика: Переименуйте слой *Layer 1* в *Фон* и добавьте на него фоновое изображение back.jpg (используйте команду меню **File—Import to Stage**).



Если посмотреть на панель **Library** в правом нижнем углу экрана, вы заметите, что в библиотеке появился новый рисунок (элемент типа *Bitmap*). Кроме того, рисунок сразу добавлен на сцену.


Практика: Чтобы защитить этот слой от случайных изменений, щелкните по черной точке ниже символа  напротив названия слоя.

Практика: Создайте новый слой с именем *Мяч*. Чтобы фон не мешал нам, отключите видимость слоя *Фон*, щелкнув на точке под значком  напротив его названия.


Создание символа

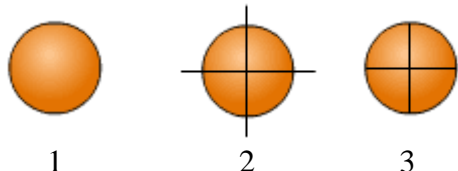
Теперь будем рисовать баскетбольный мяч.




Практика : Выберите инструмент  «овал» и отключите режим  «рисование объектов». Установите черный контур толщиной 1 пиксель и оранжевый цвет заливки. Нарисуйте круг (мяч), удерживая клавишу **Shift**.

Практика : Выделите заливку и замените ее (с помощью панели **Color**) на радиальный градиент (**Radial**) от светло-оранжевого к темно-оранжевому. С помощью инструмента  (**Gradient Transform**, клавиша **F**) переместите центр градиента немного выше и левее центра мяча (рисунок 1).

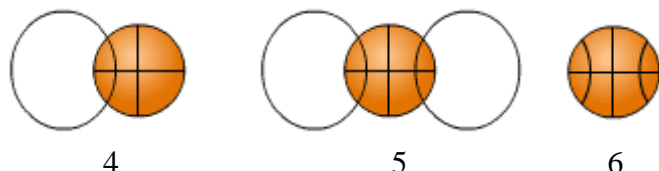
Практика : Проведите вертикальную и горизонтальную линии через центр мяча (рисунок 2).

При нажатой клавише **Ctrl** (которая временно включает инструмент ) выделите «хвостики», выступающие за мяч, и удалите их (рисунок 3).



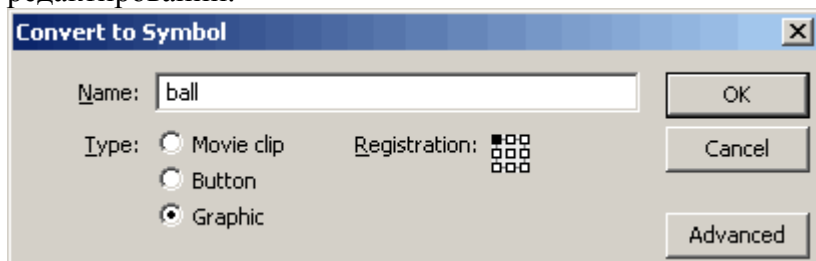
Практика : Включите снова инструмент  и режим  (рисование объектов). Отключите заливку и нарисуйте овал (рисунок 4). Включив инструмент , перетащите копию этого овал вправо (при нажатой клавише **Alt**). Выровняйте оба овала (рисунок 5).

Практика : Преобразуйте оба овала в простые контуры с помощью команды **Break Apart** из контекстного меню (клавиши **Ctrl+B**). Затем выделите и удалите все части, выступающие за контур мяча (рисунок 6).



Практика : Чтобы преобразовать нарисованный мяч в символ, выделите его инструментом  и нажмите клавишу **F8** (меню **Modify—Covert to Symbol**).

На экране появится окно, в котором нужно ввести название символа (мы введем *ball*), выбрать тип символа (*Graphic*) и определить, где находится так называемая **точка регистрации (Registration)**. Эта точка, которая определяет координаты символа на экране (они используются при управлении объектом из программы). По умолчанию точка регистрации располагается в левом верхнем углу, но это можно изменить как при создании символа (щелкнув мышкой по другому квадратику в поле **Registration**), так и потом при редактировании.



Теперь символ появился в библиотеке и его можно использовать много раз, не опасаясь увеличения объема файла. На сцене новый символ обведен голубой рамкой, крестик показывает точку регистрации.

Для создания символа «с нуля» можно использовать и другой способ: нажать клавиши **Ctrl+F8** (меню **Insert—New Symbol**). После этого в библиотеке создается пустой символ и сразу открывается в режиме редактирования.

Важно: Для того, чтобы разбить символ на отдельные составляющие, используют команду **Break Apart** из контекстного меню или из меню **Modify**.

Редактирование символа

Для редактирования символа нужно дважды щелкнуть в окне библиотеки по значку слева от имени символа. При этом вы увидите символ на сцене и сможете изменить его элементы. Полоска над сценой показывает, что мы работаем не с самой сценой, а с символом *ball*. Это очень похоже на то, что мы видели при редактировании нарисованных объектов в Теме 4. Чтобы вернуться к редактированию сцены, нужно щелкнуть мышью на тексте *Scene 1* или на стрелке.




Крестик, обозначающий точку регистрации символа, переместить нельзя, однако можно переместить изображение мяча на поле так, чтобы точка регистрации оказалась в другом месте.

Практика : Войдите в режим редактирования символа и сделайте так, чтобы точка регистрации оказалась в центре мяча.

Если символ уже есть на сцене, для перехода в режим редактирования можно сделать на нем двойной щелчок. Это так называемое **редактирование на месте** (также **Edit—Edit in Place** в меню).

Помните, что все сделанные изменения применяются ко всем копиям символа, которые есть в ролике.

Форму символа, расположенного на сцене, можно изменять с помощью инструмента . Эти изменения влияют только на тот символ, с которым вы работаете.

Практика: Отрегулируйте размер мяча так, чтобы он соответствовал размеру площадки.

6.2. Анимация движения

Практика : Сделайте видимым фоновый слой и расположите мяч в верхней части поля. В слое *Мяч* вставьте два ключевых кадра в кадры 10 и 20 (клавиша **F6**).

Заметим, что при этом фоновое изображение в кадрах 2-20 исчезло, поскольку рабочая часть временной шкалы для слоя *Фон* содержит только 1 кадр.

Практика : Выберите кадр 20 слоя *Фон* и добавьте в него новый простой кадр (клавиша **F5**).

Практика : Перейдите в кадр 10 слоя *Мяч*, выделите мяч и передвиньте его вертикально вниз почти до границы сцены. Чтобы случайно не сдвинуть его влево или вправо, лучше использовать для перемещения стрелку «вниз». При нажатой клавише **Shift** скорость перемещения увеличивается.

Теперь программа сможет автоматически построить все промежуточные кадры анимации между кадрами 1—10 и 10—20. Это **анимация движения**, поскольку форма объекта не меняется.

Практика : Щелкните правой кнопкой мыши по кадру 1 слоя *Мяч* и выберите пункт меню **Create Motion Tween** (создать анимацию движения). Сделайте то же самое для кадра 10. Просмотрите ролик.

Видим, что мяч летит неестественно — с постоянной скоростью и как будто отталкивается от верхней точки.

По законам физики, мяч летит вниз ускоренно, а вверх — замедленно. В программе *Flash* это можно сделать с помощью параметра *Ease* на панели **Properties**. Кнопка **Edit** справа от поля **Ease** позволяет точно настраивать кривую изменения скорости анимации. Обратите внимание, что в списке **Tween** теперь выбрано значение **Motion** (движение).

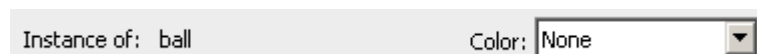
Практика : Перейдите в кадр 1 слоя *Мяч* и установите значение **Ease** равное **-100**. В кадре 10 установите этот параметр равным **100**. Просмотрите результат.

Стало уже похоже на правду. Не хватает только звука.

Практика: Импортируйте в библиотеку звуковой файл bounce.wav (меню **File—Import—Import to Library**). Добавьте новый слой и назовите его *Звук*. На этом слое вставьте новый ключевой кадр в кадр 10, выделите его и перетащите звук из библиотеки. Просмотрите ролик.

6.3. Изменение символа при анимации

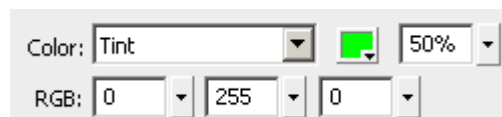
Теперь посмотрим, какие свойства символа можно менять. Если выделить мяч и посмотреть на панель **Properties**, мы увидим, что выделенный объект — копия символа *ball* из библиотеки.



Список **Color** позволяет изменять цвет данного экземпляра (эти изменения не отражаются на символе в библиотеке и на других копиях):

- **None** — нет изменений;
- **Brightness** — яркость;
- **Tint** — цветовой оттенок;
- **Alpha** — прозрачность;
- **Advanced** — экспертная настройка цвета.

Практика : в кадре 10 выделите мяч, выберите в списке **Color** вариант **Tint** (оттенок) и установите для смешивания зеленый цвет в соотношении 50%. Просмотрите ролик.



Теперь мяч «зеленеет» при полете к полу.


Сейчас мы никак не учитываем, что в момент удара мяч меняет форму («сплющивается»). Если просто уменьшить высоту символа в кадре 10, то мяч начнет «сплющиваться» с самого верха, а в самом деле это не так — быстрое изменение формы идет только при ударе. Чтобы смоделировать изменение формы мяча в момент удара, мы добавим дополнительные кадры анимации.

Практика : Выделите кадры 10-20 слоя *Мяч* и перетащите их по временной шкале вправо на 4 кадра. Скопируйте (при нажатой клавише **Alt**) кадр 14 в кадры 10 и 12. Добавьте новые

промежуточные кадры в слои *Фон* и *Звук* так, чтобы длина временных шкал всех слоев совпадали.

Если посмотреть ролик, теперь в кадрах 10-14 мяч просто стоит на месте. Мы сделаем так, что он будет при этом **менять форму**.

Так как при построении анимации программа ориентируется на положение **центра вращения**, он не должен смещаться, пока мяч находится в нижней точке. Это значит, что его надо переместить на середину нижней границы символа.

Практика : Включите инструмент  (клавиша **Q**). Перейдите в кадр 10 слоя *Мяч* и перетащите белый кружок (центр вращения) на середину нижней границы символа. Сделайте то же самое для кадра 14.


Практика : Перейдите в кадр 12 слоя *Мяч* (он должен быть ключевым!). Немного уменьшите высоту мяча, передвинув его верхнюю границу. Просмотрите ролик.

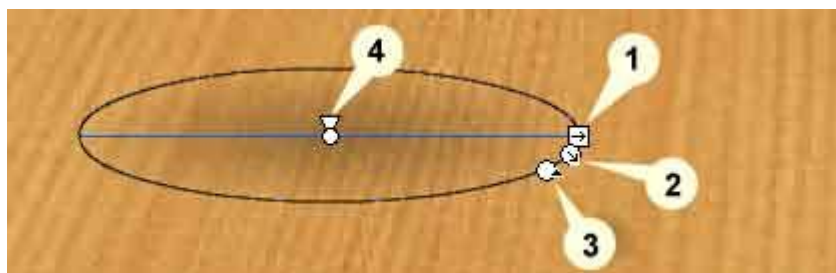
Практика : Для того, чтобы мяч двигался не слишком медленно, увеличьте частоту смены кадров до 24 и просмотрите ролик.

Теперь добавим **тень**, падающую от мяча. Будем предполагать, что осветительные лампы расположены сверху, поэтому

- тень падает вертикально вниз от мяча;
- самая темная и резкая тень — в нижней точке;
- при подъеме мяча тень размывается и увеличивается.

Для создания тени будем использовать полупрозрачные овалы, залитые черным цветом.

Практика : Добавьте слой *Тень* между слоями *Фон* и *Мяч*. Перейдите на кадр 1 и нарисуйте овал черного цвета без контура. Выделите эту заливку и на панели **Colors** установите радиальный градиент, оба цвета — черные, но у левого движка (в центре) параметр **Alpha** равен **30%**, а у правого — **Alpha=0%** (заливка полностью прозрачна). С помощью инструмента  **Gradient Transform** настройте градиент так, как показано на рисунке.



Напомним, что с помощью движка 1 можно менять соотношение сторон градиента, движок 2 регулирует масштаб, а движок 3 позволяет вращать градиент. Точка 4 обозначает центр градиента и может перемещаться.

Теперь построим тень для остальных кадров:

- тень в кадре 24 точно такая же, как и в кадре 1;
- самая темная и самая маленькая тень — в кадрах 10-14 (мячик в нижней точке);
- для изменения тени между кадрами 1-10 и 15-24 используем анимацию формы.

Практика : Перейдите в кадр 24 слоя *Тень* и вставьте новый ключевой кадр (**F6**).

Практика : Вставьте новый ключевой кадр в кадр 10 этого же слоя (клавиша **F6**). Уменьшите овал, изображающий тень от мяча в момент касания пола. Измените заливку: используйте градиент с переходом от **Alpha=90%** через **Alpha=60%** (в середине) до **Alpha=0%**. Для всех движков устанавливается черный цвет.


Практика : Включите анимацию формы для кадров 1 и 14 слоя *Тень*. Просмотрите ролик.

6.4. Направляющие


Теперь изменим ролик так, чтобы мяч вылетел слева, ударился о пол и улетел вправо за экран.

Практика : Перейдите на кадр 1 слоя *Мяч*. Выделите сразу мяч и тень (надо обвести их мышкой при нажатой клавише **Ctrl**) и переместите мяч влево за пределы поля. В кадре 24 таким же способом переместите мяч с тенью за правую границу.

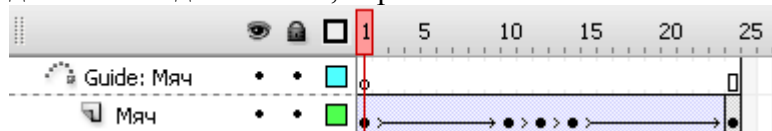
Теперь надо переместить точку удара мяча в середину сцены. Чтобы мяч в нижней точке не «дергался», желательно перемещать все изображения мяча и тени в кадрах 10-14 за один раз. Для этого можно использовать уже изученный раньше режим *редактирования нескольких соседних кадров*.



Практика : Перейдите в кадр 12 и включите режим редактирования соседних кадров, щелкнув на кнопке  под временной шкалой. Установите движки в верхней части шкалы так, чтобы они захватывали кадры 10-14. Выделите мышкой изображение мяча и тени и перетащите в середину сцены. Уменьшите частоту кадров до 12 и просмотрите ролик.

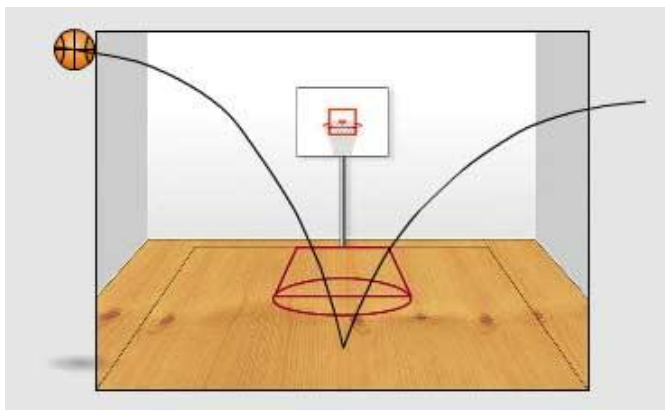
Хорошо видно, что на каждом интервале анимации мяч летит **по кратчайшему расстоянию** между начальной и конечной точками. В реальной ситуации это не так: траектория полета мяча — это кривая линия (парабола). К счастью, в программе есть возможность задать нестандартную траекторию полета с помощью слоя направляющих.

Практика: Выделите слой *Мяч* и щелкните по кнопке  (**Add Motion Guide**, создать слой направляющих) ниже списка слоев.

После этого будет создан слой особого типа с именем **Guide: Мяч**. Все, что нарисовано на этом слое, не отображается на экране. Но при анимации объекты слоя *Мяч* двигаются вдоль линий, нарисованных на слое **Guide: Мяч**.



Практика : Перейдите в кадр 1 слоя *Guide: Мяч*. Включите инструмент  и проверьте, чтобы режим рисования объектов был отключен (кнопка  в нижней части панели инструментов не должна быть нажата). Нарисуйте кривую примерно так, как показано на рисунке ниже.



Обратите внимание, что мяч в ключевом кадре сам «перескакивает» на направляющую (на ней должен находиться центр вращения).

Практика : В кадрах 10, 12 и 14 установите мяч так, чтобы центр вращения совпал с самой нижней точкой траектории. В кадре 24 также переместите мяч на траекторию. Просмотрите ролик.

6.5. Вращение

Реальный баскетбольный мяч всегда вращается в полете. В программе *Flash* есть возможность добавить вращение объекта на целое число оборотов на каждом интервале анимации движения. На панели **Properties** есть список **Rotation**, в котором можно выбрать

- **None** — нет вращения;
- **Auto** — строится автоматически, учитывая угол поворота в начальном и конечном положениях;

- **CW** (*clockwise*) — по часовой стрелке;
- **CCW** (*counter-clockwise*) — против часовой стрелки.



В поле перед словом *times* можно ввести количество оборотов.

Важно : Вращение на угол менее 360° выполняется автоматически.

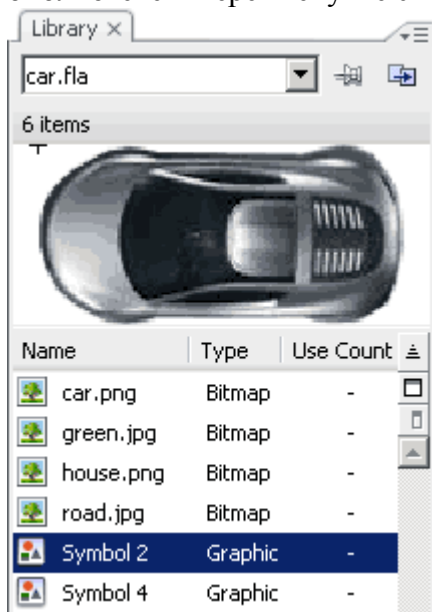
Практика : Выделите кадр 1 слоя *Мяч* и добавьте к анимации вращение — 2 оборота по часовой стрелке. То же самое сделайте для кадра 14. Чтобы лучше увидеть результат, уменьшите частоту кадров до 5. После этого можно закрыть файл.

6.6. Растровые рисунки

Мы уже говорили, что большая часть рисунков в программе *Flash* — векторные, они состоят из контуров и заливок, форма которых задается расположением узлов и касательными. Однако в фильмах можно использовать и растровые (точечные) рисунки. В предыдущем примере мы вставляли точечный фоновый рисунок, теперь мы изучим работу с изображениями более детально.

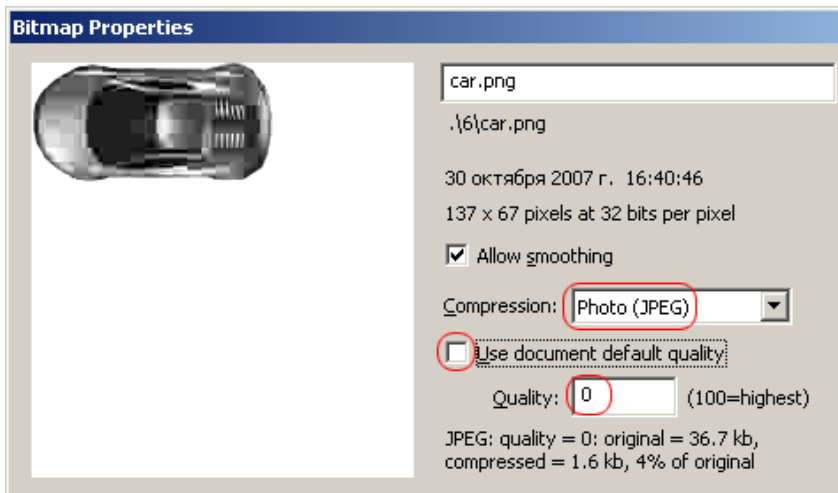
Практика : Создайте новый документ и сохраните его в папке PRACTICE\6\road.fla. С помощью команды **File—Import—Import to Library** добавьте в библиотеку рисунки car.png, house.png, green.jpg и road.jpg. Заметьте, что можно добавить несколько рисунков за один раз, выделив их в окне выбора файлов при нажатой клавише **Ctrl**.

Теперь посмотрите на окно библиотеки: кроме четырех рисунков в ней появились два **символа** типа *Graphic*, которые программа автоматически создала из рисунков в формате **PNG**. Чтобы разобраться, где какой рисунок, щелкните мышкой на этих строчках в библиотеке и переименуйте символы в *Машина* и *Дом*.



Если нажать правую кнопку мыши на рисунке в библиотеке и выбрать в контекстном меню команду **Properties** (свойства), можно увидеть свойства рисунка (размеры, цветовой режим) и изменить метод и степень сжатия в формате JPEG. Чем сильнее сжатие рисунка, тем меньше его размер, но хуже качество. Поэтому на практике приходится выбирать что-то среднее.

Практика : Щелкните правой кнопкой мыши по рисунку car.png в библиотеке и выберите пункт **Properties** (свойства) из контекстного меню. В окне **Compression** выберите вариант **Photo (JPEG)**, отключите флажок **Use document default quality** (использовать качество по умолчанию) и введите в поле **Quality** (качество) значение **0**.




Что же мы выиграли? Если посмотреть на информацию в нижней части окна, исходный размер файла (*original*) — 36,7 Кб, а размер сжатого файла (*compressed*) — 1,6 Кб, таким образом мы сжали рисунок в 25 раз. Если таким образом оптимизировать все рисунки, размер SWF-файла можно существенно уменьшить, что важно, если ролик размещается в Интернете.

Конечно, качество рисунка стало значительно хуже (появились квадраты, характерные для формата JPEG). Однако машина на сцене довольно маленькая и движется быстро, поэтому это ухудшение качества никто не заметит.




Важно, что сжатие используется только при создании SWF-файла, а рисунок в библиотеке FLA-файла не меняется (проверьте это!). Это значит, что можно всегда вернуться в окно свойств и улучшить качество, увеличив значение **Quality**.


Если для какого-то рисунка качество не задано явно (флажок **Use document default quality** включен), при создании фильма используется качество, установленное в окне настроек публикации (меню **File—Public Settings**, вкладка **Flash**, параметр **JPEG Quality**).

Практика : Включите инструмент , и нарисуйте прямоугольник **без рамки**, который закрывает всю сцену. Переименуйте слой *Layer 1* в *Фон*.


Следующий этап — залить прямоугольник рисунком-узором, который загружен в библиотеку под именем **green.jpg**.

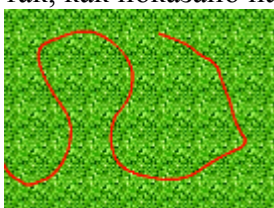
Практика : Перетащите рисунок **green.jpg** из библиотеки на свободное место рядом со сценой (чтобы не перекрыть прямоугольник). Преобразуйте рисунок в заливку, выбрав команду **Break Apart** из контекстного меню или из меню **Modify**.

Практика: Щелкните по кнопке выбора цвета заливки  , затем включите инструмент  (**Eyedropper**, пипетка) и щелкните по рисунку с травой.


При этом автоматически включится инструмент , позволяющий изменять заливку фигуры, причем заливка будет выполняться узором, по которому мы только что щелкнули.

Практика : Щелкните мышью по прямоугольнику, чтобы залить его узором. Удалите квадратик с образцом заливки и заблокируйте слой *Фон*.

Практика : Создайте новый слой и назовите его *Дорога*. Установите красный цвет линии и толщину 2. С помощью инструмента , нарисуйте красным цветом линию дороги примерно так, как показано на рисунке ниже. Не подводите линию близко к границам поля.



Практика : Создайте новый слой *Путь* и скопируйте на него только что нарисованный контур (скопировать — **Ctrl+C**, вставить в то же место — **Ctrl+Shift+V**). Пока отключите видимость слоя *Путь*.

Практика : Выделите весь путь на слое *Дорога* двойным щелчком при включенном инструменте . На панели **Properties** увеличьте толщину линии до 50. Преобразуйте контур в заливку, выбрав команду **Modify—Shape—Convert Lines to Fills**. Залейте дорогу узором, который хранится в библиотеке под именем **road.jpg**.

Практика : Перетащите домик (символ *Дом* из библиотеки) на слой *Дорога* и поставьте его в правый верхний угол сцены рядом с дорогой.

Практика : Создайте новый слой с именем *Машина*, расположите его выше всех слоев и перетащите на него символ *Машина* из библиотеки. Поставьте машину на дорогу на левый край сцены, уменьшите ее так, чтобы она поместилась на дорогу, и разверните вдоль дороги по направлению к дому.

Чтобы развернуть машину более точно, иногда приходится снимать флажок **View—Snapping—Snap to Objects**, отменяя привязку к существующим объектам.

6.7. Ориентация вдоль пути

Практика : Вставьте новый ключевой кадр в кадр 70 на слое *Машина*. Вставьте промежуточные кадры в кадр 70 всех остальных слоев (изображение в них не меняется).

Практика : На слое *Машина* в кадре 70 расположите машину на дороге около дома и разверните в нужном направлении (как она должна приехать). Добавьте анимацию движения к кадру 1 слоя *Машина*. Просмотрите ролик.

Конечно, машина движется неправильно. Надо связать ее движение со слоем направляющих. На этот раз мы не будем создавать новый слой, а используем слой *Путь*, который как раз и содержит контур пути.

Практика : Перетащите слой *Путь* на самый верх в списке слоев (прямо над слоем *Машина*). Нажмите правой кнопкой мыши на названии слоя *Путь* и выберите вариант **Guide** (направляющая).

Практика : Нажмите правой кнопкой мыши на названии слоя *Машина*, выберите **Properties** (свойства) и отметьте вариант **Guided** (направляемый слой). Просмотрите ролик.

Теперь машина движется по нужной траектории, но неправильно — она движется то боком, то задом. Конечно, можно было поставить между конечными точками еще несколько ключевых кадров, в которых развернуть машину правильно. Однако можно одним щелчком заставить машину принимать нужное направление на всей траектории.

Практика : Выделите первый кадр слоя *Машина* и отметьте флажки **Orient to Path** (ориентировать относительно пути) и **Snap** (привязать объект к пути) на панели **Properties**. Чтобы исключить дополнительное вращение, установите в окне **Rotation** вариант **None** (нет вращения). Снова посмотрите ролик.

Наконец, сделаем так, чтобы машина немного постояла у дома перед тем, как начнется новый цикл проигрывания ролика.


Практика : Выделите мышкой кадр 100 **во всех слоях** и добавьте промежуточный кадр (клавиша **F5**). Просмотрите окончательный результата.



Задание для работы : Студент рисует свою машину, дорогу, газон, дом. Создает анимацию движения машины по траектории петляющей дороги.

Пример: для заочников: **Практика :** Создаем новый слой «Фон» и рисуем фон.

Практика : Создаем новый слой «Дорога» и рисуем серпантин дороги (петляющую дорогу).

Практика : Создаем новый слой «Путь» и рисуем с помощью инструмента  линию пути, совпадающую с серединой дороги и повторяющей все повороты дороги.

Практика : Создаем символ Graphic «Машина», и рисуем машину вид сверху.

Практика : Создайте новый слой с именем *Машина* и перетаскиваем на него символ *Машина* из библиотеки. Поставьте машину на дорогу на правый край сцены, уменьшите ее так, чтобы она поместилась на дорогу, и разверните вдоль дороги по направлению к дому.

Чтобы развернуть машину более точно, иногда приходится снимать флажок **View—Snapping—Snap to Objects**, отменяя привязку к существующим объектам.

Практика : Вставьте новый ключевой кадр в кадр 70 на слое *Машина*. Вставьте промежуточные кадры в кадр 70 всех остальных слоев (изображение в них не меняется).

Практика : На слое *Машина* в кадре 70 расположите машину на дороге около дома и разверните в нужном направлении (как она должна приехать). Добавьте анимацию движения к кадру 1 слоя *Машина*. Просмотрите ролик.

Конечно, машина двигается неправильно. Надо связать ее движение со слоем направляющих. На этот раз мы не будем создавать новый слой, а используем слой *Путь*, который как раз и содержит контур пути.

Практика : Перетащите слой **Путь** на самый верх в списке слоев (прямо над слоем *Машина*). Нажмите правой кнопкой мыши на названии слоя *Путь* и выберите вариант **Guide** (направляющая).

Практика : Нажмите правой кнопкой мыши на названии слоя *Машина*, выберите **Properties** (свойства) и отметьте вариант **Guided** (направляемый слой). Просмотрите ролик.

Теперь машина движется по нужной траектории, но неправильно — она движется то боком, то задом. Конечно, можно было поставить между конечными точками еще несколько ключевых кадров, в которых развернуть машину правильно. Однако можно одним щелчком заставить машину принимать нужное направление на всей траектории.

Практика : Выделите первый кадр слоя *Машина* и отметьте флажки **Orient to Path** (ориентировать относительно пути) и **Snap** (привязать объект к пути) на панели **Properties**. Чтобы исключить дополнительное вращение, установите в окне **Rotation** вариант **None** (нет вращения). Снова посмотрите ролик.

Наконец, сделаем так, чтобы машина немного постояла у дома перед тем, как начнется новый цикл проигрывания ролика.

Практика : Выделите мышкой кадр 100 **во всех слоях** и добавьте промежуточный кадр (клавиша **F5**). Просмотрите окончательный результат.

6.8. Вложенная анимация


Сделаем фильм, где автомобиль движется по траектории, поэтому нужно использовать слой направляющих. Кроме того, при подъеме в горку движение замедляется (мы изучим два способа для достижения такого эффекта).

Наконец, самое главное: колеса автомобиля вращаются! Для создания такой анимации нам нужно создать символ *Машина*, составной частью которого будут два одинаковых (вложенных) клипа *Колесо*, которые имитируют вращающиеся колеса.

Таким образом, **один клип расположен внутри другого**, причем каждый из них имеет свою временную шкалу: для клипа *Машина* установлена анимация движения в виде движения по заданной траектории, клип *Колесо* обеспечивает вложенную анимацию — вращение колес.

Практика : Создайте новый документ, установите размеры поля 700 на 300 пикселей.

Практика : Переименуйте слой *Layer 1* в *Фон* и нарисуйте фоновый рисунок. Заблокируйте слой от изменений.

Практика : Создайте новый слой *Путь* и нарисуйте на нем с помощью инструмента  траекторию движения машины, как показано на рисунке ниже.



Практика : Создайте новый символ *Машина* (клавиши **Ctrl-F8**), выберите тип символа **Movie Clip**. Нарисуйте корпус машины .

Для того, чтобы колеса вращались, нам нужно вместо простого графического символа *Колесо* использовать **клип** (символ типа **Movie Clip**). После этого мы добавим анимацию (вращение) **внутри** этого клипа.

Практика : Создайте новый символ (**Ctrl+F8**) типа **Movie Clip** (клип) с именем *Колесо*. Нарисуйте на поле клипа *Колесо* так, чтобы центр вращения оказался в том же месте, что и крестик (точка регистрации).

Практика : В кадре 30 клипа *Колесо* вставьте новый ключевой кадр и включите анимацию движения для кадра 1. На панели **Properties** установите вращение (**Rotation**) по часовой стрелке (**CW**) на 2 оборота (**2 times**). Просмотрите результат.

Практика : Откройте клип *Машина* в режиме редактирования. Выберите из списка клип *Колесо*. Состыкуйте Колесо с корпусом. Таким же образом вставьте второе колесо.

Практика : Вернитесь к редактированию сцены. Создайте слой *Машина* и добавьте в кадр 1 машину из библиотеки так, чтобы она оказалась на красной линии слева от видимой области.

Практика : Сделайте так, чтобы слой *Путь* стал слоем направляющих для слоя *Машина*. Для этого надо в свойствах слоя *Путь* (правая кнопка мыши — **Properties**) выбрать вариант **Guide**, а в свойствах слоя *Машина* установить **Guided**.

Практика : В кадре 80 слоя *Машина* вставьте новый ключевой кадр (**F6**) и расположите машину справа от видимой области так, чтобы ее центр вращения (белый кружок) оказался на направляющей линии. В кадре 80 остальных слоев вставьте новые промежуточные кадры (**F5**).

Практика : Для кадра 1 слоя *Машина* включите анимацию движения и отметьте флажки **Orient to Path** и **Snap** на панели **Properties**. Просмотрите фильм.

6.9. Изменение скорости анимации

Остается замедлить ход машины при движении в горку и добавить звук. Обычно при анимации движения объект равномерно движется из начального положения в конечное, т.е. скорость не изменяется. Если надо сделать неравномерное движение, используют два способа:

1. весь интервал анимации разбивается на **несколько подынтервалов** (в середину добавляется несколько ключевых кадров); для каждого интервала задается своя скорость анимации, которая определяется начальным и конечным положением объекта;
2. настраивается **кривая перемещения** вдоль траектории, которая вызывается при нажатии кнопки **Edit** на панели **Properties**

Практика : Перемещая считывающую головку над временной шкалой, найдите положение, когда машина начинает ехать в гору (около кадра 40), и немного сместите машину (программа вставит новый ключевой кадр). Теперь перетащите мышью этот ключевой кадр в кадр 20 и просмотрите результат.

Практика : Отмените предыдущие операции, чтобы скорость снова стала постоянной.

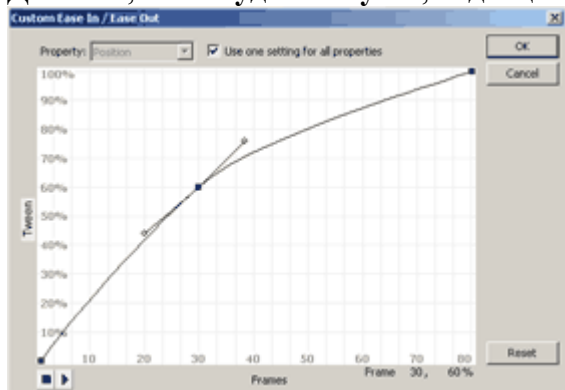
Теперь применим второй способ. Если перейти в кадр, где включена анимация движения, и щелкнуть по кнопке **Edit** в панели **Properties**, мы увидим линию, которая задает скорость движения вдоль траектории. На оси абсцисс отмечены номера кадров, а на оси ординат — процент пройденного пути.

Сначала эта линия прямая, т.е. скорость движения постоянна. Если нужно изменить это поведение, щелчком мышки добавьте **новый узел** на линию и переместите в новое положение.

Узел имеет две **касательные**, которыми можно регулировать углы входа и выхода линии.

Направляющие есть также и у конечных точек.

Для того, чтобы **удалить узел**, надо щелкнуть по нему при нажатой клавише **Alt**.



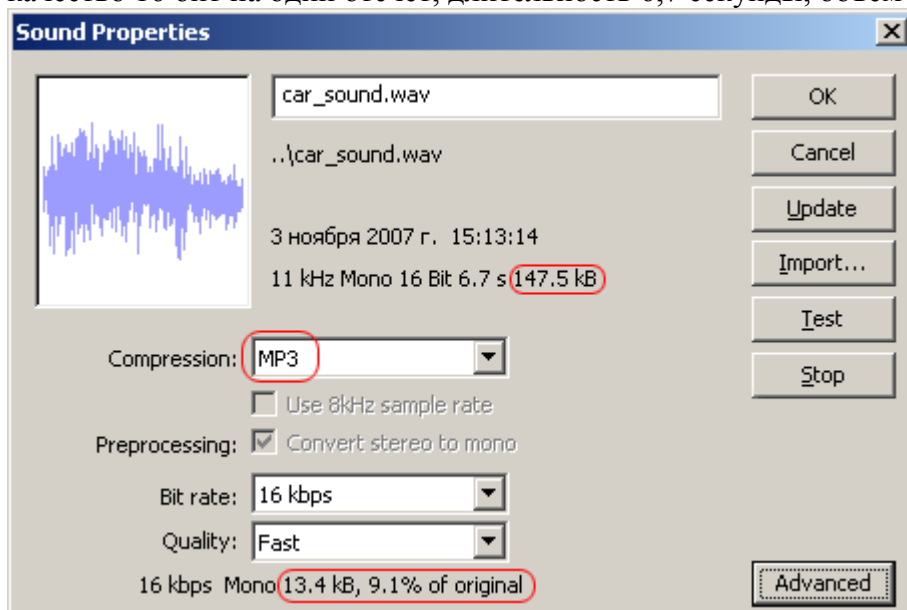
Практика : Перейдите в кадр 1 слоя *Машина* и измените кривую примерно так, как на рисунке выше. Просмотрите фильм.

Практика : Добавьте новый слой *Звук* и перетащите в кадр 1 звук *car_sound.wav* из библиотеки.

Файлы в формате **WAV**, как правило, имеют большие размеры, поскольку не используют сжатие. Посмотрим, можно ли уменьшить звуковой файл.

Практика : Щелкните правой кнопкой мыши на звуке *car_sound.wav* и выберите пункт **Properties** из контекстного меню.

В верхней части окна мы увидим свойства файла: частота дискретизации 11 кГц, качество 16 бит на один отсчет, длительность 6,7 секунды, объем **147,5 Кб**.



Практика : В списке **Compression** (сжатие) выберите алгоритм **MP3**, оставьте все параметры по умолчанию.

Ниже окна **Quality** мы видим, что сжатый звук занимает всего 13,4 Кб, то есть 9,1% от исходного объема. С помощью алгоритма **MP3** мы сжали его более, чем в 10 раз.

Важно, что звук сжимается только при создании SWF-файла, а в библиотеке хранится в исходном виде. Это значит, что всегда можно безболезненно изменить степень сжатия. Конечно, сделать качество лучше, чем в исходном варианте, не удастся. Если сжатие звука не задано явно, используются настройки окна **File—Publish Settings** (вкладка **Flash**).

Практика : Нажмите кнопку **ОК**, чтобы принять этот вариант.

Чтобы движение автомобиля выглядело более натурально, добавим падающую тень. Учитывая, что в горах днем солнце стоит достаточно высоко, сделаем небольшую тень в виде полупрозрачной черной заливки.

Практика : Откройте символ *Машина* в режиме редактирования. Добавьте новый слой *Тень* и расположите его ниже основного слоя *Layer 1*. Нарисуйте тень в виде заливки черного цвета с прозрачностью 50%. Для того, чтобы было удобнее работать с тенью, основной слой с изображением машины можно временно сделать невидимым. Просмотрите окончательный вариант и сохраните его.

Задание для работы : Студент рисует свою машину, пейзаж. Создает анимацию движения машины по траектории. При подъеме в горку движение автомобиля замедляется. *Колеса автомобиля должны вращаться*. Для создания такой анимации нужно создать символ *Машина*, составной частью которого будут два одинаковых (вложенных) клипа *Колесо*, которые имитируют вращающиеся колеса.


6.10. Анимация текста

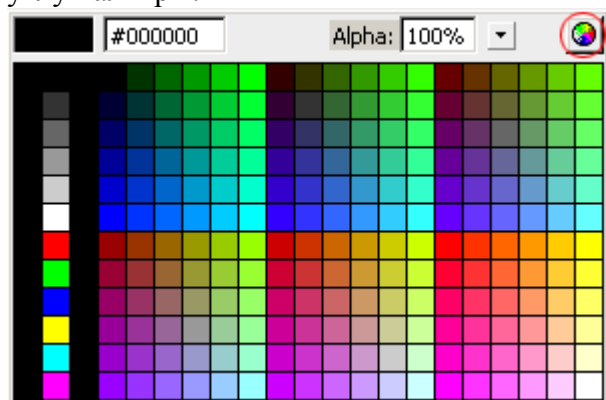
Изменение размера

Как известно, в полночь с 31 декабря на 1 января Новый Год «вылезает» из часов Спасской башни Кремля. Теперь мы сделаем аналогичный эффект с помощью *Flash*

Практика : Создайте новый документ размером 300 на 300 пикселей и сохраните его в папке PRACTICE\6 под именем clock fla.

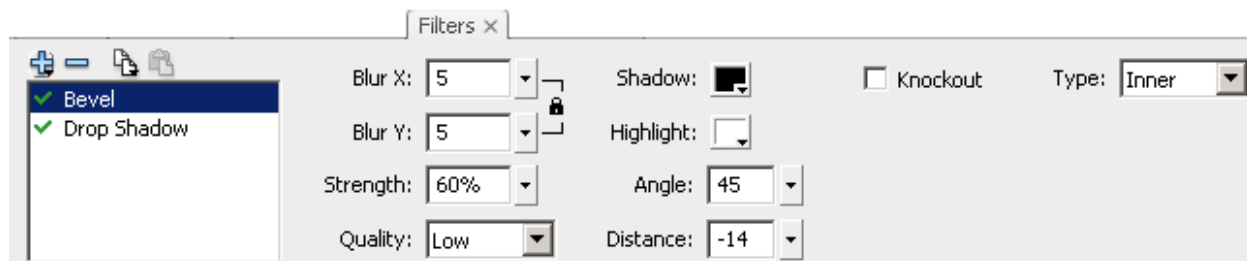
Практика : Переименуйте слой *Layer 1* в *Фон* и добавьте рисунок clock.jpg. Заблокируйте слой *Фон*.

Практика : Создайте слой *Текст* и поместите на него текст **2019**: шрифт *Arial*, жирный, размер 96, дополнительный интервал между буквами 4, цвет **R=255** (красный), **G=229** (зеленый), **B=172** (синий). Для точного ввода цвета используйте кнопку  в правом верхнем углу палитры.



Практика : Примените к тексту фильтры **Drop Shadow** и **Bevel**. Для фильтра **Bevel** установите специальные параметры:

- тип (*Type*): **Inner** (внутренний);
- сила (*Strength*): **40%**;
- дистанция (*Distance*): **-14**.



Практика : Преобразуйте текст в символ с именем *2019* (клавиша **F8**).

Теперь построим анимацию, в ходе которой текст будет увеличиваться, начиная из центра циферблата. Отведем на это увеличение 6 секунд (72 кадра при частоте 12 кадров в секунду), а затем зафиксируем надпись еще на 3 секунды.

Практика : На слое *Текст* вставьте в кадры 72 ключевой кадр. Затем добавьте в кадр 108 обоим слоям промежуточный кадр

Практика : Перейдите в кадр 1 слоя *Текст* и уменьшите текст, расположив его в центре циферблата. Включите анимацию движения для кадра 1 и просмотрите результат.

Легко заметить, что циферблат во второй половине фильма мешает смотреть надпись, отвлекает внимание. Поэтому мы перекроем его черным прямоугольником, для которого будем изменять параметр **Alpha** от **0%** (полностью прозрачен) до **80%**.

Практика : Создайте слой *Затенение* между фоном и текстом и нарисуйте на нем черный прямоугольник, перекрывающий всю сцену. Добавьте в этом слое ключевой кадр в кадр 72. В кадре 1 установите с помощью панели **Color** прозрачный цвет (**Alpha=0**), а в кадре 72 — **Alpha=80%** (почти непрозрачный). Включите для кадра 1 анимацию формы (**Shape Tween**).

Последний штрих — остается добавить звук курантов. Звуковой файл хранит запись одного удара, которая занимает около 3 секунд. Мы поместим в фильм 3 копии этого звука (в кадры 1, 36 и 72).

Практика : Создайте новый слой *Звук* и вставьте ключевые кадры в кадрах 36 и 72. Загрузите в библиотеку файл *kuranty.mp3* и перетащите звук из библиотеки в кадры 1, 36 и 72. Просмотрите результат.

Побуквенная анимация

Разбив текст на отдельные буквы, можно строить интересные анимационные эффекты.

Практика : Создайте новый документ, установите размеры поля 400 на 300 пикселей. Загрузите фоновый рисунок *dawn.jpg* и заблокируйте слой. Сохраните файл под именем *luga.n fla*.

Практика : Создайте новый слой *Текст*, введите в нижней части сцены текст *Лугань* (шрифт *Arial*, черный, жирный, размер 70) и выровняйте его по середине сцены (панель **Align**, включить режим **To Stage**).

Теперь мы разобьем текст на отдельные буквы и разместим каждую из них на отдельном слое.

Практика : Выделите текст и примените команды **Break Apart** и **Distribute to Layers** из контекстного меню. Удалите слой *Текст* (он теперь пустой).

Программа создала несколько новых слоев, их имена совпадают с буквами текста.

Практика : Выделите кадр 10 во всех слоях с буквами и добавьте новый ключевой кадр (**F6**). Затем выделите кадр 1 во всех слоях-буквах (выделятся все буквы) и клавишей «вправо» переместите буквы за правую границу сцены (нажатие клавиши **Shift** ускоряет процесс).

Практика : Включите анимацию движения для всех слоев-букв, вставьте новый промежуточный кадр в кадр 10 слоя *Фон* и просмотрите результат.

Если посмотреть на библиотеку, можно заметить, что в нее добавлены новые символы, созданные автоматически из букв. Это необходимо программе для построения анимации движения.

Практика : Выделите кадры 1-10 слоя, где находится буква у, и перетащите вправо, так чтобы анимация начиналась с кадра 6. Сдвиньте и другие слои с буквами так же, на 5 кадров от предыдущей буквы. Выделите кадр 35 всех слоев-букв и вставьте новый ключевой кадр, в тот же кадр фонового слоя вставьте промежуточный кадр. Просмотрите результат.

Далее, используя фильтр **Bevel**, сделаем так, чтобы буквы постепенно становились золотистыми. Однако применить фильтры можно только к тексту или символам типа **Movie Clip**. Сейчас буквы имеют тип **Graphic**, это можно увидеть на панели **Properties**. Поэтому нужно будет изменить тип символов на **Movie Clip**.

Практика : Перейдите в кадр 35, выделите все буквы и на панели **Properties** измените тип символа на **Movie Clip**.

Сейчас все готово для создания анимации, которая сводится к постепенному применению фильтра.

Практика : Выделите кадр 50 во всех слоях-буквах и вставьте новый ключевой кадр. Не снимая выделения с букв, перейдите на панель **Filters** и примените фильтр **Bevel**, изменив некоторые параметры:

- цвет тени (*Shadow*): **R=116, G=89, B=78**;
- цвет светлых областей (*Highlight*): **R=253, G=202, B=139**;
- дистанция (*Distance*): **-6**.

Добавьте анимацию движения ко всем слоям с буквами в кадре 35.

Остается сделать небольшую паузу перед началом нового цикла.

Практика : Вставьте промежуточные кадры в кадр 70 во всех слоях и просмотрите результат. Сохраните файл.

Вопрос для самостоятельной работы.

1. Что такое Символ?
2. Какие три типа символов различают в программе *Flash* ?
3. Как задать нестандартную траекторию движения объекта?

Термины: Символы, слои, типы текста

Литература:[[1](#) С. 100-120,168-189; [2](#) С.277-324]

Тема 7. Знакомство с ActionScript

7.1. Зачем нужен ActionScript?

Вся анимация, о которой мы говорили до этого, работала независимо от зрителя, которые не мог вмешиваться в ход процесса. Говорят, что при этом отсутствует **интерактивность** — взаимодействие между человеком и программой.

Чтобы человек мог управлять ходом развития событий, нужно научиться принимать информацию от пользователя и изменять состояние объектов на экране. Для этого надо написать программу — набор команд на языке *ActionScript*.

Язык *ActionScript* относится к группе скриптовых языков и очень похож на *JavaScript*. Он использует многие правила грамматики, заимствованные из языка *Cu*.

С помощью языка *ActionScript* можно

- управлять ходом проигрывания фильма;
- создавать анимацию программным способом, без использования временной шкалы;
- принимать данные от пользователя;
- загружать данные из файлов;
- управлять звуками;
- ... и многое другое.

В версии *Adobe Flash 9.0 (CS3)* основным языком программирования является *ActionScript 3.0*, однако он достаточно сложен для изучения на начальном этапе и мы не будем его рассматривать.

Вместе с тем поддерживается и старый «простой» вариант — язык **ActionScript 2.0**, который и будет использоваться далее.

Программа на *ActionScript* называется **сценарием**. Она состоит из отдельных блоков кода, которые могут быть связаны с некоторыми элементами фильма:

- **ключевыми кадрами**, код выполняется каждый раз, когда этот кадр проигрывается;
- **кнопками** или **клипами**, код выполняется при возникновении какого-нибудь **события** (щелчок по кнопке, загрузка клипа, перетаскивание мышью, нажатие клавиши).

7.2. Переменные

Объявление

Переменная — это величина, имеющая имя и значение. Переменные объявляются с помощью слова `var`:

```
var x = 12, y;
```

Здесь введено две переменных с именами `x` и `y`, в переменную `x` записано значение 12, а переменная `y` не определена, то есть команда трассировки

```
trace ( y );
```

выдаст результат `undefined` (значение не определено). Такой же результат выдает и команда `trace (z);`

потому что переменная `z` вообще неизвестна. Для того, чтобы отличить существующую переменную от неизвестной, можно записать в нее специальное нулевое значение `null`:

```
var y = null;
```

Если тип переменной явно не указан, она может принимать любые значения. Например:

```
var x = 1; // число
```

```
x = "Ку-ку!"; // строка
```

```
x = false; // логическое значение
```

Однако при объявлении лучше явно указывать тип переменной. Это позволяет обнаруживать многие ошибки еще до выполнения программы. Существует три простых типа:

- `Number` — число;
- `String` — строка;
- `Boolean` — логическое значение.

Тип переменной указывается после ее имени через двоеточие

```
var x:Number = 0,
```

```
    y:String = "qq",
```

```
    b:Boolean = false;
```

В переменные типа `String` можно записывать символьные строки, заключенные в кавычки или одиночные апострофы:

```
var s1:String = "qq1", s2:String = 'qq2';
```

Логические переменные принимают только два значения: `true` (истина) и `false` (ложь):

```
var b:Boolean = false;
```

```
b = true;
```

```
b = (a < 2);
```

В последнем случае значение переменной `b` будет равно `true`, если условие справа от знака равенства верно.

Если в переменную попытаться записать значение неверного типа, вы получите сообщение об ошибке сразу при трансляции программы (то есть, при переводе ее в машинные коды), а не во время выполнения. Например, такой код вызывает ошибку:

```
var x:Number = 1;
```

```
x = "Ку-ку!";
```

Видимость переменных

Различают три вида переменных:

- **переменные временной шкалы** (или переменные клипа) — объявляются в коде, связанном с каким-то кадром;
- **локальные переменные** — объявляются внутри *функций* для временного хранения данных;
- **глобальные переменные** — известны всем функциям.

Глобальные переменные объявляются с помощью описателя `_global`:

```
_global.x = 12;
```

Обратите внимание, что слово `var` здесь использовать не нужно, такие переменные рассматриваются как свойства объекта `_global`. К переменной `x`, которая объявлена выше, можно обращаться из любой функции и из кода любого клипа просто по имени.

Если в области видимости есть несколько переменных с одним и тем же именем, сначала ищется локальная переменная, затем — переменная текущего клипа, и только потом — глобальная переменная.

Переменные других клипов «не видны», для обращения к ним нужно указывать явно клип-родитель:

```
mc.x = 1;
```

```
_root.x = 12;
```

```
_parent.x = 123;
```

Присваивание

Для присваивания переменной нового значения используется знак `=`. Слева от него пишут имя переменной, а справа — выражение:

```
a = 4*(c + 2) + 3 / (r - 4*w) + d % 3;
```

Знак `*` обозначает умножение, знак `/` — деление, а `%` — взятие остатка от деления.

В выражении арифметические операции выполняются в следующем порядке:

- действия в скобках;
- умножение, деление и взятие остатка (слева направо);
- сложение и вычитание (слева направо).

Этот порядок называется **приоритетом** (старшинством) арифметических операций.

Символьные строки можно «сцеплять» с помощью оператора `+`:

```
po = 20;
```

```
s = "Вася " + "пошёл гулять.";
```

```
qq = "Объект" + po;
```

Если в выражении участвуют разнотипные данные, происходит автоматическое преобразование к одному типу. Так в последней строке в переменную `qq` записывается строка `Объект20`.

Очень часто в программах используют операторы `++` (*инкремент*, увеличение переменной на 1), и `--` (*декремент*, уменьшение переменной на 1). Операторы

```
i ++; k --;
```

означают то же самое, что

```
i = i + 1; k = k - 1;
```

Существует также сокращенная запись арифметических операций:

```
a += 20; b -= c - d; c *= a + b; d /= 2 * c; f %= 12;
```

Этот код можно заменить на такие операторы в «нормальной» форме:

```
a = a + 20;
```

```
b = b - (c - d);
```

```
c = c * (a + b);
```

```
d = d / (2 * c);
```

```
f = f % 12
```

Объекты

Объект — это нечто, имеющее свойства и методы. В среде *Flash* существуют встроенные объекты (например, `Array`, `MovieClip`, `Key`). Кроме того, можно строить свои объекты:

```
var car = new Object();
```

```
car.v = 10;
```

```
car.year = 1998;
```

В среде *Flash* можно использовать объектно-ориентированное программирование, то есть создавать свои классы объектов, наделять их свойствами и методами.

Главная особенность объектов — это так называемая ссылочная адресация. То есть, при объявлении

```
var obj = new Object();
```


переменная `obj` хранит не сам объект, а только его **адрес** (ссылку на объект). Поэтому оператор присваивания

```
obj2 = obj;
```

не создает в памяти новый объект, который является копией объекта `obj`, а просто копирует адрес первого объекта в переменную `obj2`. После этого `obj` и `obj2` указывают на один объект. Если мы хотим действительно построить копию объекта, адрес которого хранится в `obj`, можно сделать так:

```
var obj2 = new Object();
```

```
for ( prop in obj ) obj2[prop] = obj[prop];
```

Здесь в цикле перебираются все свойства первого объекта и копируются во второй.

Переменная `prop` (символьная строка) — это название очередного свойства. Запись `obj[prop]` означает «свойство объекта `obj`, название которого хранится в `prop`».

7.3. Массивы

Основные понятия

Массив — это набор элементов, имеющих общее имя. Каждый элемент имеет собственный номер, начиная с **нуля**. Для обращения к элементу массива используются квадратные скобки.

Важно: Нумерация элементов массива с **нуля** непривычна для обычных людей, но широко используется в программировании (языки *C#, JavaScript, Java, ActionScript*).

В отличие от многих языков программирования, в одном массиве могут быть разнотипные элементы: целые числа, вещественные числа, строки, логические переменные.

Массив из трех элементов можно объявить и заполнить так:

```
A = new Array(3);
```

```
A[0] = 12;
```

```
A[1] = 4.56;
```

```
A[2] = "Ку-ку!";
```

или так:

```
A = new Array (12, 4.56, "Ку-ку!");
```

или так:

```
A = [12, 4.56, "Ку-ку!"];
```

Копирование

При работе с массивами, как и с другими объектами (кроме чисел, строк и логических переменных), есть один тонкий момент. Рассмотрим пример кода:

```
A = [1, 2, 3];
```

```
B = A;
```

```
B[1] = 99;
```

```
trace(A);
```

Здесь создается массив `A`, затем он копируется в массив `B`. Потом изменяется `B[1]` и массив `A` выводится в окно **Output**. Вроде бы массив `A` не должен измениться, однако мы увидим в окне **Output** строчку:

```
1,99,3
```

Почему же изменился массив `A`? Дело в том, что оператор `B=A` НЕ создает новый массив, а просто копирует в `B` **адрес** массива `A`, то есть `A` и `B` обращаются к одной области памяти.

Поэтому, изменив `B`, мы изменили и `A`. Чтобы действительно создать копию массива, нужно заменить оператор `B=A` на код

```
B = new Array();
```

```
for(i=0; i<A.length; i++) B[i] = A[i];
```

Здесь используется встроенное свойство `length` объекта `Array` — длина массива.

Многомерные массивы

Элементом массива может быть другой массив:

```
A = new Array();
```

```
A[0] = [1, 2, 3];
```

```
A[1] = [4, "Вася"];
```

```
A[2] = [true, "qq", 4];
```

В таком (**двухмерном**) массиве каждый элемент имеет два индекса. Например, A[1][0] имеет значение 4, а A[2][1] — это символьная строка qq.

Методы

Для работы с массивами можно использовать следующие встроенные методы класса Array:

- **объединение** массивов, в результате строится новый массив:
C = A.concat (B, 12);
в массиве C записаны все элементы массива A, затем все элементы массива B, а затем — число 12;
- **добавление элементов** в начало массива
A.unshift (3, 4, "qq");
и в конец массива
A.push (3, 4, "qq");
- **удаление элемента** с начала массива
x = A.shift();
и с конца массива
x = A.pop();
удаленный элемент записывается в переменную x;
- **перестановка элементов массива в обратном порядке:**
A.reverse();
- **извлечение** части массива в новый массив:
B = A.slice (n1, n2);
здесь в массив B копируются все элементы массива A с номерами от n1 до n2 включительно;
- **сортировка** массива
A.sort();
По умолчанию элементы массива сортируются по возрастанию в алфавитном порядке; если надо сортировать числа, используется форма \

```
A.sort ( Array.NUMERIC );
```

Для сортировки по убыванию нужно указать еще параметр Array.DESENDING:

```
A.sort ( Array.NUMERIC + Array.DESENDING );
```

Если нужна нестандартная сортировка, в качестве параметра указывается ссылка на функцию, которая принимает два параметра (элемента массива) и возвращает

- отрицательное число, если первый «меньше» второго;
- 0, если элементы равны;
- положительное число, если первый «больше» второго.

Например:

```
A = [12, 328, 731];
```

```
A.sort ( cmpFun );
```

```
function cmpFun(a, b) {  
  return (a%10) - (b%10);  
}
```

```
trace ( A );
```

Массив будет отсортирован по возрастанию последней цифры:

```
731,12,328
```

потому что a%10 — это остаток от деления числа a на 10 или его последняя цифра.

7.4. Условные операторы

Условные операторы используются для выбора одного из двух вариантов. **Полная форма** оператора:

```
if ( условие ) {
  ... // блок if
} else {
  ... // блок else
}
```

Если *условие* истинно, выполняется *блок if*, а если ложно — *блок else*. В каждом из них можно размещать любые операторы. в том числе и другие условные операторы.

Второй блок (начиная со слова *else*) может отсутствовать. При этом если условие не выполняется, никаких действий не будет. Такая форма называется **неполной формой** условного оператора:

```
if ( условие ) {
  ... // блок if
}
```

Если внутри фигурных скобок стоит всего один оператор, скобки можно не ставить.

Простейшие условия — это отношения

- <, <=, >, >= — меньше, меньше или равно, больше, больше или равно;
- == равно;
- != не равно.

Например,

```
if ( x < 0 ) {
  trace ( "x отрицательный" );
} else {
  if ( x == 0 ) {
    trace ( "x равен 0" );
  } else {
    trace ( "x положительный" );
  }
}
```

Здесь используется **вложенный условный оператор** (один внутри другого).

Сложные условия образуются из простых (отношений) с помощью **логических операций**:

- ! — «не» (отрицание, обратное условие);
- && — «и» (одновременное выполнение условий);
- || — «или» (требуется выполнение хотя бы одного из условий).

Операции выполняются в следующем порядке:

- действия в скобках;
- операция «не»;
- отношения;
- операция «и»;
- операция «или».

Этот порядок называется **приоритетом** (старшинством) логических операций. Например

```
a = 1; b = 2; c = 3;
```

```
if ( a < b && a < c )
  trace ( "a - наименьшее" );
if ( b > a || b > c )
  trace ( "b - не наименьшее" );
```

Оба условных оператора здесь сработают и в окне **Output** будут выведены оба сообщения.

7.5. Циклы

Цикл — это многократное повторение последовательности действий. Существуют циклы с известным числом шагов и циклы с условием.

Цикл с переменной

Цикл с известным числом шагов (или цикл с переменной) имеет вид

```
for ( начальные значения; условие; изменения ) {
  ... // тело цикла
}
```

```
}
```

В заголовке цикла три части, разделенные точками с запятой. Операторы в первой части выполняются один раз при входе в цикл.

В цикле выполняются все операторы, расположенные внутри фигурных скобок (в теле цикла). Если там всего один оператор, скобки можно не ставить. Цикл работает до тех пор, пока **условие** в заголовке цикла не станет ложным.

После каждого шага цикла выполняются операторы, записанные в третьей части (*изменения*).

В цикле

```
for ( i=0; i<10; i++ )  
  trace ( i );
```

в окно **Output** выводятся в столбик числа от 0 до 9 включительно. Действительно:

- при входе в цикл выполняется оператор `i=0`;
- в конце каждого шага цикла переменная `i` увеличивается на 1 (оператор `i++`);
- цикл выполняется пока `i<10`, то есть, закончится при `i=10` (это значение не будет выведено).

Переменная `i` в таком цикле называется **переменной цикла**.

Цикл с переменной часто используют для обработки массивов, при этом `i` обозначает номер элемента:

```
A = [1, 2, 3];  
for ( i=0; i<A.length; i++ )  
  A[i] *= 2;
```

Этот цикл умножает все элементы массива `A` на 2.

Циклы с условием

Цикл с условием начинается ключевым словом `while`, после которого в скобках записывают условие продолжения работы цикла:

```
while ( условие ) {  
  ...  
}
```

Цикл заканчивается, когда условие становится ложным. Например:

```
i = 0;  
while ( i < 10 ) {  
  trace ( i );  
  i ++;  
}
```

Этот цикл, так же, как и в предыдущем примере, выводит числа от 0 до 9.

Если условие всегда истинно, программа *заиклиивается* (будет работать бесконечно долго).

Например, часто забывают изменять переменную в теле цикла:

```
i = 0;  
while ( i < 10 ) {  
  trace ( i ); // заиклиивание!!!  
}
```

Проверка условия стоит в начале цикла (**цикл с предусловием**), поэтому он ни разу не выполняется, если условие неверно при входе в цикл.

Если в теле цикла стоит один оператор, фигурные скобки можно не ставить.

Цикл с постусловием отличается тем, что проверка условия выполняется **после** очередного шага:

```
do {  
  ...  
}  
while ( условие );
```

Такой цикл всегда выполняется хотя бы один раз.

Перечисление свойств

Особый вид цикла используется для перечисления всех свойств объекта:

```
for ( var переменная in объект ) {
```

```
...
}
```

В этом цикле *переменная* «проходит» все свойства *объекта*, ее значение на каждом шаге — название очередного свойства. Например, цикл:

```
var obj = new Object();
obj.x = 10;
obj.y = 20;
for ( var prop in obj ) {
    trace ( prop + " = " + obj[prop] );
}
```

выведет в окно **Output** названия свойств, x и y, и их значения:

```
x = 10
y = 20
```

Управление циклом

Для изменения порядка выполнения цикла используются два оператора:

- `break` — досрочный выход из цикла;
- `continue` — переход в конец цикла.

Например, цикл

```
for ( i=0; i<10; i++ ) {
    trace ( i );
    if ( i >= 3 ) break;
}
```

выполнится только 4 раза.

Другой цикл

```
for ( i=0; i<10; i++ ) {
    if ( i >= 3 ) continue;
    trace ( i );
}
```

выполнится 10 раз, но в окно **Output** будут выведены только числа 0, 1 и 2.

7.6. Функции

Функция — это блок кода, который можно использовать многократно (вызывать).

Преимущество функций состоит в том, что

- не нужно несколько раз писать одинаковый код, рискуя ошибиться при переписывании;
- сокращается длина программы;
- при отладке изменения нужно вносить только в одно место.

Функции объявляются с помощью слова `function`:

```
function qq() {
    ...
}
```

Имя этой функции — `qq`, при ее вызове `qq()`; выполняются все команды внутри фигурных скобок. Многоточие здесь и далее означает любой код на *ActionScript*.

Функция может принимать **параметры** или **аргументы** — дополнительные данные, которые влияют на ее работу. Параметры перечисляются через запятую в скобках:

```
function qq ( x ) { ... }
function ww ( x:Number, s:String ) { ... }
```

Функция `qq` принимает один параметр, его тип может быть любой. Функция `ww` принимает два параметра, причем их типы жестко заданы: первый — число, второй — символьная строка.

Функция может возвращать результат (число, строку, логическое значение и т.д.) с помощью оператора `return`. Например:

```
function Sum ( x, y ) {
    var s = x + y;
```

```
return s; // или return x + y;
}
```

Чтобы использовать это значение, надо записать его в переменную:

```
w = Sum ( 2, 3 );
trace ( w );
```

Типы функций

Функции, так же, как и переменные, бывают трех типов:

- **функции временной шкалы** (функции клипа), которые связаны с клипом и могут рассматриваться как его методы;
- **глобальные функции**, которые можно использовать без указания адреса из любого места в программе;
- **локальные функции**, которые объявляются внутри другой функции.

Чаще всего используют функции клипов. Например, если в одном из кадров главной временной шкалы объявлена функция

```
function qq () { ... }
```

с другой временной шкалы (из обработчика другого клипа) она вызывается как `_root.qq()`.

Ту же функцию можно объявить по-другому:

```
qq = function() { ... }
```

При этом фактически создается переменная типа «ссылка на функцию» и в нее записывается адрес кода. Разница (в сравнении с первым способом) состоит в том, что в переменную `qq` можно потом записать адрес другой функции:

```
qq = function () { trace(1); }
```

```
qq();
```

```
qq = newFunc;
```

```
qq();
```

```
function newFunc () { trace ( 2 ); }
```

Если в первый раз при выполнении оператора `qq()` вызывается первая (безымянная) функция, то второй раз — функция `newFunc`. Заметим, что после присваивания `qq=newFunc` к первой функции уже никак не обратиться — ее адрес потерян.

Глобальные функции объявляются с помощью слова `_global`:

```
_global.qq = function ( x ) { ... }
```

Эту функцию можно вызывать просто как `qq()` из любого места программы, в том числе из обработчиков событий клипов.

Локальные функции объявляются внутри другой функции и известны только там. В примере

```
trace(b);
```

```
a();
```

```
function a() {
```

```
  function b() {
```

```
    trace(b);
```

```
  }
```

функция `b` — локальная для функции `a`, поэтому первый вызов `trace(b)` даст результат `undefined` (не определено), а второй (изнутри функции `a`) — `[type Function]`, то есть, найдена функция с таким именем.

Обработчики событий

Обработчики событий — это специальные методы кнопок и клипов. Например, вместо определения обработчика события `on(release)` можно записать адрес нужной функции в свойство `onRelease` кнопки или клипа.

Пусть на текущем монтажном столе есть кнопка с именем `btn` (имя кнопки задается в левой части на панели **Properties**). Разберем такой код:

```
btn.onRelease = function () {
```

```
  trace("q");
```

```
  btn.onRelease = second;
```

```
}
```



```
function second() {
    trace("qq");
    btn.onRelease = null;
}
```

Вначале обработчик события кнопки — это безымянная функция, которая выводит на экран букву q и, кроме этого, меняет обработчик события на адрес функции second.

При втором щелчке вызывается функция second, которая выводит две буквы q и уничтожает обработчик, записывая в него null. На последующие щелчки кнопка не будет реагировать.

7.7. Управление проигрыванием. Кнопки

Задача для работы: В Теме 6 Студент рисовал свою машину, пейзаж. Создавал анимацию движения машины по траектории. При подъеме в горку движение автомобиля замедлялось. *Колеса автомобиля должны вращаться..* Теперь нужно добавить кнопки управления роликом. **Добавить кнопки «Стоп», «Пуск», «Перейти в начало».** Машина должна **останавливаться перед подъемом.**

Команды управления

Чаще всего используются следующие команды управления:

- stop(); — остановить проигрывание;
- play(); — продолжить проигрывание с текущего места;
- gotoAndStop(...); — перейти на заданный кадр и остановиться, в скобках указывают номер кадра или его метку (имя);
- gotoAndPlay(...); — перейти на заданный кадр и продолжить проигрывание.

С помощью этих команд и кнопок мы построим ролик.

Кнопки в нижней части поля будут иметь следующий смысл

- ◀◀ перейти в начальное положение
- ▶ начать (продолжить) движение
- остановить движение

Практика : Откройте файл ship fla из папки PRACTICE\7 и просмотрите ролик.

Практика : Перейдите в кадр 15 слайда *Судно* и нажмите клавишу **F9**, чтобы вывести на экран панель **Actions**. Введите код:

```
stop();
```

Просмотрите результат.

Для того, чтобы продолжить проигрывание фильма, мы добавим на сцену кнопку — символ специального типа.

Кнопки

При создании символа можно выбрать тип **Button** (кнопка). Так называется клип, имеющий 4 кадра:

- **Up** — обычное состояние кнопки;
- **Over** — вид кнопки, когда на нее наведен курсор мыши;
- **Down** — нажатая кнопка;
- **Hit** — область реагирования.

В каждом из них могут быть любые рисунки. Более того, можно создавать многослойные изображения и добавлять звук для каждого из состояний.

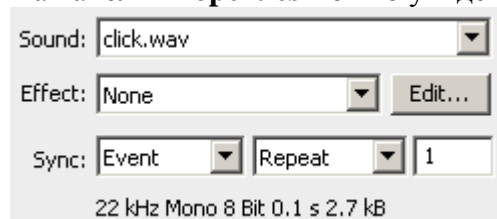
Практика : Создайте новый символ (клавиши **Ctrl+F8**) типа **Button** (кнопка) и назовите его *Кнопка_Старт*. Нарисуйте примерно вот такую кнопку: ▶ Вставьте новый ключевой кадр в кадре *Over* и измените цвет стрелки на светло-зеленый. Затем вставьте ключевой кадр в кадре *Down* и измените цвет стрелки на красный.

Мы не задали никакого изображения в кадре *Hit*, это значит, что областью реагирования является вся непрозрачная область последнего кадра *Down*.

Теперь добавим звук, который будет сигнализировать о нажатии кнопки.

Практика : Добавьте новый слой *Звук* и вставьте новый ключевой кадр *Down*. Перетащите на поле звук **click.wav** из библиотеки.

На панели **Properties** можно увидеть свойства этого звука:



Характеристики звука видны в нижней строчке: частота дискретизации 22 кГц, монофонический (левый и правый каналы звучат одинаково), 8-битное кодирование, время звучания 0,1 сек, размер файла 2,7 Кб.

Список **Effect** (эффект) позволяет выбрать один из стандартных эффектов (затухание, переход с левого канала на правый и т.п.). Значение **None** означает, что никакой эффект не применяется. Кнопка **Edit** служит для ручной настройки эффектов.

Параметр **Sync** для кнопок должен принимать значение **Event** (событие). Правее **Repeat 1** означает повторение 1 раз. Кроме **Repeat** возможен еще вариант **Loop** (зацикливание, бесконечное повторение).

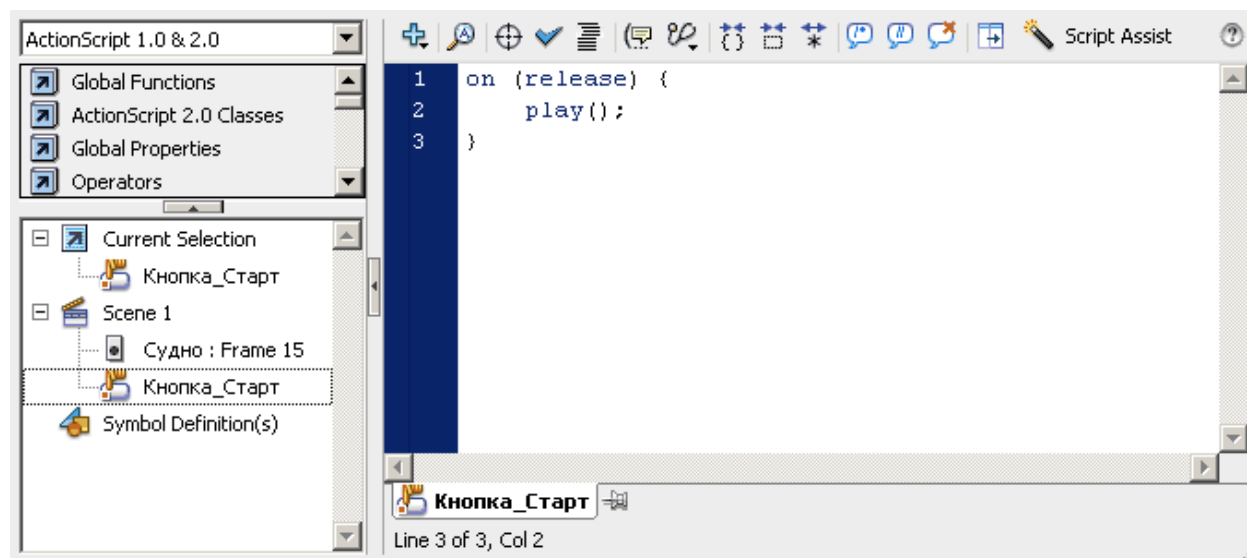
Практика : Вернитесь к сцене. Создайте новый слой *Кнопки*, выделите кадр 1, перетащите кнопку из библиотеки в нижнюю часть поля и отрегулируйте ее размер.


Реакции на события


Чтобы кнопка «заработала», нужно определить ее реакцию на нажатие. Если выделить кнопку и нажать клавишу **F9**, на экране появляется панель **Actions** (действия), где можно написать программный код, связанный с кнопкой. Вот пример такого кода:

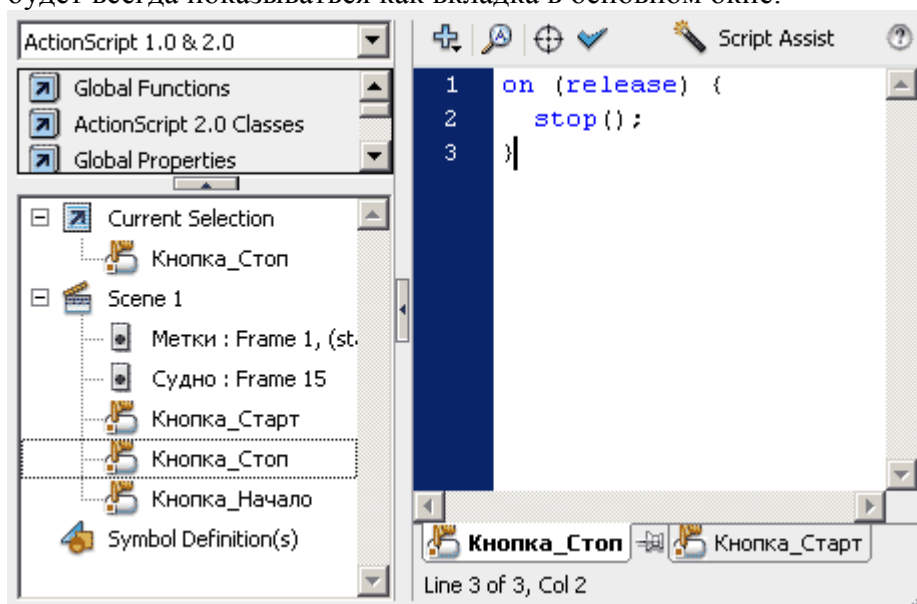
```
on ( release ) {  
    play();  
}
```

Слово **on** начинает обработчик события, **release** (*освобождение*) в скобках указывает само событие — отпускание мыши над кнопкой. Далее внутри фигурных скобок записывают команды, которые нужно выполнить: в данном случае — продолжить проигрывание фильма. Язык *ActionScript* использует правила записи команд, очень похожие на язык Си. Каждая команда заканчивается точкой с запятой.



Нажав на кнопку  (или клавиши **Ctrl+T**), можно проверить правильность записи кода (соответствие правилам языка *ActionScript*). Если вы увидели сообщение "*The script contains*



no errors.", ошибок нет. Сообщения об ошибках выводятся в специальном окне **Compiler Errors** (оно появляется автоматически или может быть вызвано нажатием клавиш **Alt+F2**). Кнопка  позволяет закрепить активное окно с кодом. Дело в том, что код может быть разбросан по нескольким местам (кадрам, кнопкам, клипам). В активном окне мы видим тот блок, который выделен в левой части панели **Actions**. Если прикрепить какой-то блок, он будет всегда показываться как вкладка в основном окне:



На рисунке активное окно — это код, связанный с кнопкой *Стоп*, а код кнопки *Стоп* был прикреплен и показывается в виде вкладки.

Практика : Выделите кнопку и добавьте в окне **Actions** код

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

Щелкните по кнопке , чтобы проверить правильность набранного кода, а затем — по кнопке  для автоматического выравнивания.

В левой части панели **Actions** находится справочная информация (*Toolbox*). Сверху — списки функций и операторов языка (их можно вставлять в код щелчком мыши), ниже — список всех объектов, с которыми связан код *ActionScript*. Сейчас видно, что какой-то код есть в кадре 15 слоя *Судно* (*Судно: Frame 15*) и у кнопки (элементы блока *Scene 1*). В данный момент мы редактируем код кнопки (*Current Selection*). Это окно удобно использовать для быстрого перехода от одного блока кода к другому.


С помощью кнопки  можно убрать панели *Toolbox*, оставив только редактор кода.


Практика : Выделите кадр 1 и добавьте к нему код

```
stop();
```

Закройте панель *Actions* и проверьте работу фильма.

Теперь добавим еще 2 кнопки: для остановки фильма и для перехода в начало. Поскольку на них должен быть другой рисунок, в библиотеке нужно создать еще два символа-кнопки. Например, можно продублировать существующую кнопку и изменить рисунок, выбрав команду **Duplicate** из контекстного меню, которое появляется при нажатии правой кнопки мыши на названии символа в библиотеке.

Практика : Создайте кнопку  с именем *Кнопка_Стоп* для остановки проигрывания. Для этого продублируйте кнопку в библиотеке, дайте ей новое имя и исправьте рисунки во всех кадрах (*Up*, *Over*, *Down*). Добавьте новую кнопку на слой *Кнопки* и настройте так, чтобы она останавливала фильм.

Практика : Добавьте еще одну кнопку , которая возвращает фильм в начало (к кадру 1) с помощью кода

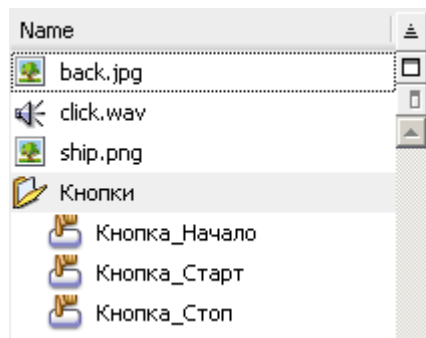
```
on (release) { gotoAndPlay(1); }
```

Проверьте ролик.




Практика : Измените команду, связанную с последней кнопкой, на `gotoAndStop(1);` и посмотрите, что изменилось.

Когда в библиотеке очень много элементов, список получается очень длинный и в нем сложно разобраться. Чтобы облегчить жизнь, сходные элементы объединяют в **папки**. Папку в библиотеке можно открывать и закрывать двойным щелчком мыши.

Практика : Выделите все кнопки в библиотеке (щелкая на них при нажатой клавише **Ctrl**) и выберите пункт **Move to New Folder** (переместить в новую папку) из контекстного меню. Дайте папке имя *Кнопки*.



Теперь надо сделать, чтобы кнопки на сцене были одинакового размера, и выровнять их. Для этого используется панель **Align** (выравнивание, клавиши **Ctrl+K**). Проверьте, чтобы режим *To Stage* был отключен (он используется для выравнивания относительно всей сцены).

Практика : Выделите все кнопки и включите панель **Align**. Сначала сделайте одинаковой высоту кнопок ( в группе *Match Size*). Затем выровняйте центры по вертикали ( в группе *Align*) и распределите равномерно по горизонтали ( в группе *Distribute*).

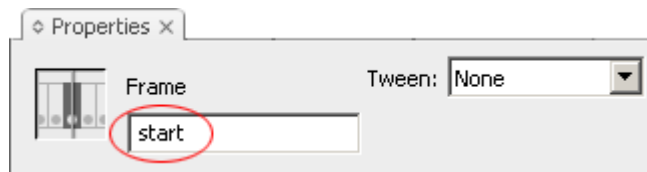
Для кнопок, так же, как и для клипов, можно применять фильтры (панель **Filters**).

Практика : Для придания кнопкам более красивого вида примените фильтр **Bevel** и настройте его параметры на ваш вкус.

Метки кадров

В ходе работы кадры часто перемещаются, поэтому переходы по номерам кадров нежелательны. Вместо этого кадру (только **ключевому!**) можно присвоить имя (**метку**) и выполнять переход по метке, а не по номеру. Для меток обычно создается отдельный слой.

Практика : Создайте новый слой *Метки* и выделите кадр 1. На панели **Properties** введите метку кадра, как показано на рисунке.



Замените команду, связанную с последней кнопкой, на `gotoAndStop ("start");` и проверьте ролик.

Обратите внимание, что на временной шкале в кадре 1 слоя *Метки* появилась метка *start*:

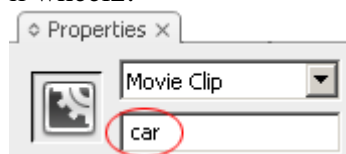


7.8. Адреса: дети, родители и корень

Задача для работы: В ролике от Темы 6, где прибавляли кнопки управления роликом - «Стоп», «Пуск», «Перейти в начало». Добавьте в код слоя *Программа* команды, останавливающие вращение колес, а в код обработчика нажатия кнопки — команды для запуска вращения колес. Сохраните файл и просмотрите результат.

Если теперь посмотреть ролик, выяснится, что машина стоит на месте, но ее колеса вращаются. Это происходит потому, что команда `stop()` остановила только проигрывание главной временной шкалы, а внутренние клипы (колеса) продолжают работать. Чтобы ответить на вопрос «Как остановить колеса?» нужно разобраться с адресами внутри флэш-ролика. Но прежде всего, дадим имена объектам на сцене, к которым мы будем обращаться: машине и двум колесам внутри нее.

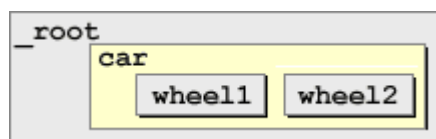
Практика : Выделите машину и на панели **Properties** введите имя этого элемента `car`. Откройте клип *Машина* для редактирования и таким же образом дайте колесам имена `wheel1` и `wheel2`.



Итак, в этом фильме 4 монтажных стола со своими временными шкалами:

- **главный** (корневой) монтажный стол, он называется `_root`;
- монтажный стол клипа **Машина**;
- два монтажных стола **для каждого колеса**.

Связи этих монтажных столов показаны на схеме:



Клип `car` находится внутри главного монтажного стола `_root`, а клипы `wheel1` и `wheel2` — внутри клипа `car`.

Главный монтажный стол `_root` является «родителем» (*parent*) для клипа `car`.

Клип `car` является «сыном» (*child*) для `_root` и родителем для клипов `wheel1` и `wheel2`.

Клипы `wheel1` и `wheel2` — сыновья для `car`, у них самих нет потомков.

Абсолютные адреса

Каждый объект имеет свой абсолютный адрес, по которому его можно найти из любого места. Например, квартира Васи Пупкина может иметь такой абсолютный адрес:

ЛНР, Луганск, ул. Оборонная, д. 5, кв. 25

Здесь на первом месте стоит страна, затем — город и т.д.

Для объектов *Flash*-фильма абсолютный адрес начинается с `_root`, так что абсолютные адреса всех клипов, показанных на схеме, выглядят так:

```
_root
_root.car
_root.car.wheel1
_root.car.wheel2
```

Как видим, для обращения к сыновьям используется точка.

Абсолютные адреса не зависят от того, внутри какого монтажного стола они используются.

Чтобы остановить проигрывание для какого-то монтажного стола, надо перед командой `stop()` поставить его адрес и точку:

```
_root.stop();
_root.car.stop();
_root.car.wheel1.stop();
```

```
_root.car.wheel2.stop();
```

Если мы применяем абсолютные адреса, нужно помнить, что они станут неверными, если машина будет находиться уже не на главном монтажном столе, а станет частью другого клипа (Вася Пупкин переехал).

Относительные адреса

Относительные адреса зависят от того, где находится объект, использующий этот адрес. Вася Пупкин может сказать, что Коля Иванов живет через 2 дома **от него**.

Для обращения к родителю используют слово `_parent`. Например, чтобы из клипа `wheel1` остановить проигрывание для монтажного стола клипа `car`, надо применить команду `_parent.stop()`;

Эта команда сработает из клипа `wheel2`, поскольку его родителем также является `car`.

Если два раза использовать `_parent`, мы обращаемся к «деду». Например, команда `_parent._parent.stop()`;

останавливает проигрывание на главном монтажном столе из клипа `wheel1` или `wheel2`.

К сыновьям можно обращаться просто по имени. Например, команда `car.stop()`;

допустима внутри `_root`. Кроме того, здесь можно было использовать и слово `this` (текущий монтажный стол):

```
this.car.stop();
```

Для остановки `wheel1` из `_root` можно применить одну из команд

```
car.wheel1.stop();
```

```
this.car.wheel1.stop();
```

Наконец, самый сложный случай. Обратиться к «брату» можно только через родителя.

Чтобы остановить второе колесо из `wheel1`, надо написать

```
_parent.wheel2.stop();
```

7.9. Звук: новые подробности

Практика : Откройте файл `car fla` из папки `PRACTICE\7`. Создайте новый слой: *Кнопка*. На слой *Кнопка* поместите кнопку из библиотеки.

Практика : В первый кадр слоя *Кнопка* добавьте код `stop()`;

Практика : Выделите машину и на панели **Properties** введите имя этого элемента `car`. Откройте клип *Машина* для редактирования и таким же образом дайте колесам имена `wheel1` и `wheel2`.

Практика : Добавьте в код кадр 1 слоя *Кнопка* команды, останавливающие вращение колес, а в код обработчика нажатия кнопки — команды для запуска вращения колес. Сохраните файл и просмотрите результат.

Практика : Создайте новый слой *Звук* и перетащите в кадр 1 звук `fastcar.wav` из библиотеки. Просмотрите результат.

Получилось, что машина стоит на месте, а звук уже слышен. Чтобы этого не случилось можно начать проигрывать звук со второго кадра.

Практика: Перетащите кадр 1 слоя *Звук* в кадр 2 и просмотрите результат.

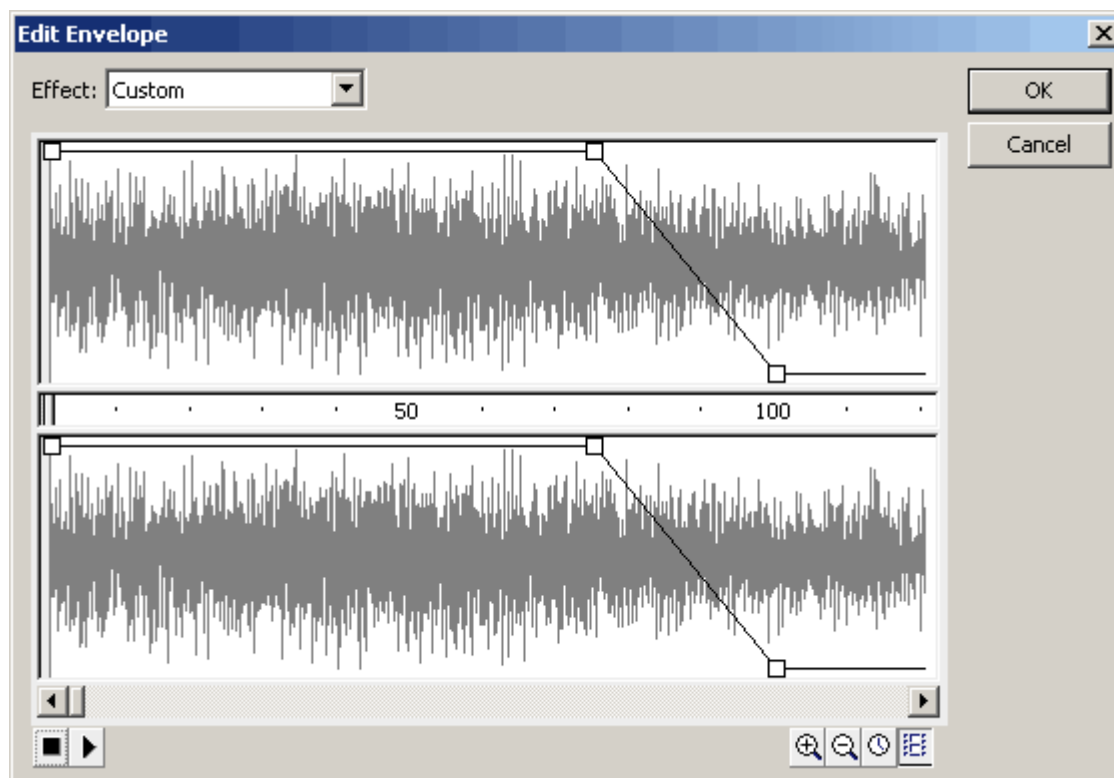
Теперь звук слышен даже тогда, когда ролик начал проигрываться с начала и машина появилась на стартовой позиции. Чтобы остановить звук, надо вставить в слое *Звук* последний ключевой кадр и выбрать в параметрах звука **Sync=Stop** (остановить звук).

Практика : Вставьте ключевой кадр в последний кадр слоя *Звук*. На панели **Properties** в списке **Sound** выберите `fastcar.wav` и установите для параметра **Sync** значение **Stop**.

Параметр **Sync** имеет еще и другие возможные значения:

- **Event** — звук начинается, когда анимация доходит до этого кадра; если звук уже играет, при повторном вхождении в кадр запускается вторая копия и смешивается с «хвостом» первой;
- **Start** — то же самое, что и **Event**, но вторая копия не запускается, если одна уже играет;
- **Stream** — звук «режется» на кусочки и каждый из них «привязывается» к кадру (это бывает необходимо, когда требуется перейти в середину анимации и проигрывать звук с нужного места, а не с начала).

Теперь звук резко обрывается, когда машина уходит из кадра. В жизни такого не бывает — затухание должно быть постепенным. Чтобы добиться этого, нужно вручную настроить громкость каналов. Если щелкнуть по кнопке **Edit** на панели **Properties**, мы увидим окно настройки **Edit Envelope** (изменить огибающую):



Верхняя диаграмма показывает звук в левом канале, нижняя — в правом. Линии с белыми квадратиками регулируют громкость. В самом начале стоит только один квадратик в начале линии, щелчками мыши можно установить еще 7 узловых точек и, перемещая их, изменить громкость (сверху — наибольшая, снизу — наименьшая).

Кнопки и позволяют изменять масштаб временной оси, на которой откладываются номера кадров (если нажата кнопка) или время в секундах (кнопка).

Кнопки и служат для проигрывания звука (с учетом примененного эффекта) и его остановки.

Практика : Откройте окно **Edit Envelope**, включите режим (номера кадров) и установите удобный масштаб временной оси. Добавив два узла к огибающим, измените их так, как показано на рисунке сверху (звук полностью затухает от 75-го до 100-го кадра). Сохраните фильм, просмотрите окончательный результат и закройте файл.

7.10 Свойства и события клипа

Задача для работы: Построить ролик в котором кнопки со стрелками на поле изменяют масштаб и угол поворота ракеты.

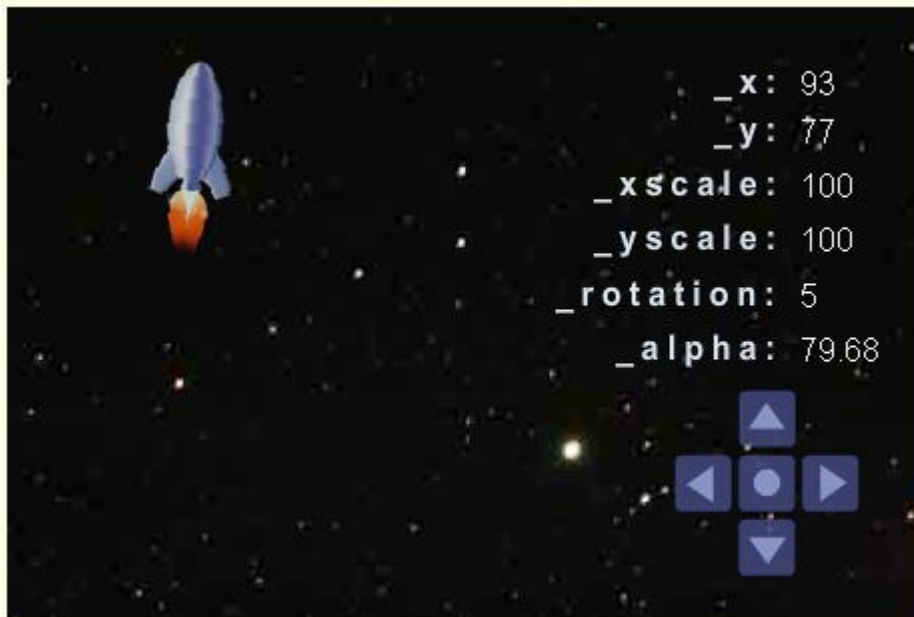
В правой части поля выводятся текущие **свойства** клипа *Ракета*:

- `_x`, `_y` — координаты точки регистрации (отсчитываются от левого верхнего угла экрана);
- `_width`, `_height` — ширина и высота клипа в пикселях;
- `_xscale`, `_yscale` — масштабы по осям **X** и **Y** (равны 100 в исходном состоянии);
- `_rotation` — угол поворота в градусах вокруг точки регистрации (равен 0 для клипа в библиотеке);
- `_alpha` — степень непрозрачности (от 0 до 100).

Свойства объектов

В этом разделе мы более подробно познакомимся с клипами и узнаем, как обрабатывать информацию от клавиатуры и мыши.

В результате будет построен ролик, показанный ниже.



Здесь ракету можно перетаскивать мышкой и перемещать клавишами-стрелками на клавиатуре. Кнопки со стрелками на поле изменяют масштаб и угол поворота ракеты. Если водить мышью над ракетой, она постепенно становится прозрачной. Чтобы опять сделать ее полностью непрозрачной, надо щелкнуть по центральной кнопке с кружком. В правой части поля выводятся текущие **свойства** клипа *Ракета*:

- `_x`, `_y` — координаты точки регистрации (отсчитываются от левого верхнего угла экрана);
- `_width`, `_height` — ширина и высота клипа в пикселях;
- `_xscale`, `_yscale` — масштабы по осям **X** и **Y** (равны 100 в исходном состоянии);
- `_rotation` — угол поворота в градусах вокруг точки регистрации (равен 0 для клипа в библиотеке);
- `_alpha` — степень непрозрачности (от 0 до 100).

Изменение свойств

Практика: Откройте файл `space.fla` из папки `PRACTICE\7`. Переместите точку регистрации клипа *Ракета* на ось симметрии ракеты (для этого надо открыть клип для редактирования и переместить изображение так, чтобы крестик оказался в нужном месте).

Практика: Добавьте слой *Кнопки* и разместите на нем 1 кнопку типа *Кнопка0* и 4 кнопки типа *Кнопка* из библиотеки. С помощью панели **Transform** (меню **Windows—Transform** или клавиши **Ctrl+T**) разверните три кнопки со стрелками на 90, 180 и 270 градусов.

Расположите кнопки ровно, так как на образце в начале раздела.

Центральная кнопка с кружком будет восстанавливать полную непрозрачность ракеты, кнопки «влево» и «вправо» вращают ее (изменяют свойство `_rotation`), а кнопки «вверх» и «вниз» — изменяют размеры (свойства `_xscale` и `_yscale`).

В языке *ActionScript* (так же, как в *Cu* и *Java*) для операций увеличения или уменьшения значений переменных на некоторую величину часто используется сокращенная запись:

```
a = a - 5;
```

```
b = b + 10;
```

выполняется точно так же, как

```
a -= 5;
```

```
b += 10;
```

Практика: Добавьте код для кнопок. Центральная кнопка:

```
on (release) {  
    rocket._alpha = 100;  
}
```

Стрелка «влево»:

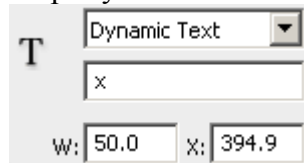
```
on (release) {  
    rocket._rotation -= 10;  
}
```

Стрелка «вправо» — то же самое, но со знаком «плюс». Стрелка «вниз»

```
on (release) {  
    rocket._xscale += 5;  
    rocket._yscale += 5;  
}
```

Стрелка «вверх» — то же самое, но со знаком «минус». Просмотрите результат и сохраните файл.

Практика: Добавьте текст `_x`: и справа от него — еще одно текстовое поле для вывода текущей координаты `_x`. Для него в панели **Properties** установите тип **Dynamic Text**, имя `x` и ширину **W=50**.



События клипа

Клипы (символы типа **Movie Clip**) могут реагировать на те же события, что и кнопки, с помощью обработчиков `on (...)`. Кроме того, есть еще несколько особых событий, которых у кнопок нет. Наиболее важные из них:

- `load` — клип полностью загружен;
- `enterFrame` — произошел переход к следующему кадру клипа;
- `mouseDown` — нажата кнопка мыши;
- `mouseMove` — мышь переместилась;
- `mouseUp` — кнопка мыши отпущена;
- `keyDown` — нажата клавиша на клавиатуре;
- `keyUp` — клавиша отпущена.

Для обработки этих событий используются обработчики `onClipEvent (...)`, которые можно добавить на панели **Actions**, предварительно выделив клип.

При частоте кадров 12, событие `enterFrame` возникает 12 раз в секунду, даже если клип содержит единственный кадр. Мы используем это событие для того, чтобы постоянно обновлять `x`-координату ракеты на экране.

Динамическое текстовое поле с именем `x` имеет свойство `text`, которое можно изменять из программы (это и есть надпись на экране). Для обращения к свойству объекта используется точка, то есть адрес текста принимает вид `x.text`. Таким образом, запись текущей `x`-координаты клипа `rocket` в текстовое поле `x` должна выглядеть так:

```
x.text = rocket._x;
```

Практика: Выделите ракету на сцене и откройте панель **Actions** (клавиша **F9**). Введите код обработчика события enterFrame:

```
onClipEvent (enterFrame) {  
    x.text = rocket._x;  
}
```

Посмотрите результат.

Вы увидели, что x-координата на экране по-прежнему равна нулю, хотя клип находится не на левой границе экрана, и значит его действительная x-координата больше нуля.

Чтобы обнаружить ошибку, мы применим **трассировку** — вывод (в дополнительное окно) отладочных сообщений, которые позволят понять, что в самом деле происходит внутри программы.

Практика: Добавьте в обработчик строчку
trace (x);

При запуске появится окно **Output** (включается также через меню **Window—Output** или клавишей **F2**), в котором появляются строчки undefined.


Происходит следующее: мы попросили программу сказать, что же такое x в этой точке, и в ответ получили, что x — это непонятно что (*undefined*, не определено). Значит, текстовое поле x не найдено — неверно задан его адрес. Сообщение появляется 12 раз в секунду, каждый раз, когда происходит событие enterFrame. Заметим, что точно такой же результат даст строчка

```
trace ( rocket );
```

Значит, обращение к клипу тоже неверно.

Дело в адресах. В обработчиках событий клипа текущим монтажным столом считается сам клип. Поэтому для того, чтобы обратиться к текстовому полю x, которое принадлежит не клипу, а главному монтажному столу, его адрес надо записать в виде `_root.x` (абсолютный адрес) или `_parent.x` (относительный адрес).

В правой части оператора надо обратиться к свойству `_x` текущего объекта, то есть написать `this._x` (относительный адрес) или `_root.rocket._x` (абсолютный адрес). Но можно то же самое написать проще, ведь `_x` обозначает то же самое, что `this._x`.

Важно: Чтобы не ошибиться при обращении к объекту, можно нажать на кнопку  и выбрать из списка существующий объект. Программа автоматически вставит его адрес, относительный (при выборе варианта **Relative**) или абсолютный (вариант **Absolute**).

Практика : Выделите ракету на сцене и откройте панель **Actions** (клавиша **F9**). Исправьте код обработчика события enterFrame:

```
onClipEvent (enterFrame) {  
    _root.x.text = _x;  
}
```

Попробуйте вставить ссылку на текстовое поле с помощью кнопки .

Практика: Добавьте аналогичные пары (статический текст и динамическое текстовое поле) для остальных свойств клипа и дополните обработчик события командами, заполняющими эти поля. Посмотрите результат.

Важно: Заметьте, что координата `_x` и другие свойства клипа — целые числа. При записи в текстовые поля они будут автоматически преобразованы в символьный вид.

Значения свойств клипа можно не только читать, но и изменять. После загрузки клипа в память мы установим его начальные координаты в обработчике события load.

Практика: Добавьте в код клипа *Ракета* обработчик события load (клип загружен в память):

```
onClipEvent (load) {  
    _x = 100;  
    _y = 100;  
}
```

7.11 «Мышиные события»

Задача для работы: Сделать так, чтобы при прохождении курсора мыши над ракетой, ракета становилась немного более прозрачной (на 5 процентов). Сделать чтобы ракета вращалась против часовой стрелке, если мышь движется в левой половине поля, и по часовой стрелке, если в правой. Ракету можно перетаскивать мышкой и перемещать клавишами-стрелками на клавиатуре.

Теперь займемся событиями от мыши. Кнопки и клипы могут обрабатывать несколько «мышинных» событий:

- `press` — над объектом нажата кнопка мыши;
- `release` — над объектом отпущена кнопка мыши;
- `releaseOutside` — кнопка мыши отпущена вне объекта;
- `rollOver` — указатель мыши наведен на объект (без нажатия);
- `rollOut` — указатель мыши ушел с объекта (без нажатия);
- `dragOut` — указатель мыши ушел с объекта при нажатой кнопке;
- `dragOver` — при нажатой кнопке мышь уходит с объекта, а затем наводится на него снова (аналогия — «чистим ботинки»).

Мы сделаем так, чтобы при событии `rollOut` ракета становилась немного более прозрачной (на 5 процентов). Для этого надо уменьшить значение `_alpha`.

Практика : Добавьте обработчик события мыши для клипа *Ракета*:

```
on (rollOut){
  _alpha -= 5;
}
```

Заметьте, что параметр `_alpha` по смыслу не должен быть меньше нуля. Будет неплохо, если вы сможете обеспечить это условие, добавив в обработчик условный оператор `if`.

Перетаскивание

Перетаскивание мышкой тоже делается очень просто: при нажатии кнопки на объекте (событие `press`) надо «захватить» объект и начать перетаскивание:

```
startDrag ( this );
```

В скобках указывается адрес объекта (`this` означает «текущий объект», то есть клип).

При отпуске кнопки (событие `release`) надо закончить перетаскивание:

```
stopDrag();
```

Практика: Добавьте обработчики событий мыши для клипа *Ракета*:

```
on ( press ){
  startDrag(this);
}
on ( release ){
  stopDrag();
}
```

В скобках после слова `on` можно перечислять через запятую несколько событий, на которые надо реагировать одинаково. Для этого часто удобно использовать средство **Script Assist** (помощник скриптов).

Практика: Выделите строчку кода `on(release)` и щелкните по словам **Script Assist** в правом верхнем углу панели **Actions**. Отметьте флажок **releaseOutside** и закройте окно повторным щелчком по словам **Script Assist**.

Теперь при отпуске мыши за пределами клипа перетаскивание также заканчивается.

«Мышиные события» клипа

Клип, как уже говорилось, также может реагировать на события `mouseDown`, `mouseUp` и `mouseMove`. Важно, что эти обработчики вызываются не только тогда, когда мышь над клипом, а при любом изменении состояния мыши. При этом координаты курсора хранятся в свойствах `_xmouse` и `_ymouse` главного монтажного стола `_root`.

Например, мы сделаем, чтобы ракета вращалась против часовой стрелки, если мышь двигается в левой половине поля, и по часовой стрелке, если в правой. Для этого надо написать обработчик события `mouseMove`, в котором менять свойство `_rotation` клипа *Ракета*. Чтобы определить направление вращения, будем сравнивать x-координату клипа с половиной ширины окна, то есть с `_root._width/2`.

Практика: Добавьте еще один обработчик к коду клипа:

```
onClipEvent (mouseMove) {
    if (_root._xmouse < _root._width/2)
        _rotation -= 0.5;
    else _rotation += 0.5;
}
```

При просмотре обнаруживается неточность: во время перетаскивания ракета продолжает вращаться, ведь событие `mouseMove` происходит и вызывается обработчик. Чтобы прекратить вращение во время перетаскивания, надо где-то запоминать, что мы тащим ракету. Легче всего сделать это при помощи специальной *логической* переменной `dragging`, которая может принимать значения `true` («да», истина) и `false` («нет», ложь). В начале перетаскивания мы запишем в эту переменную `true`, а когда перетаскивание закончено, запишем в нее `false`.

Практика: Измените обработчики событий:

```
on ( press ){
    startDrag(this);
    dragging = true;
}
on ( release, releaseOutside ){
    stopDrag();
    dragging = false;
}
```

Теперь можно использовать значение этой переменной в обработчике `mouseMove` — вращать ракету только тогда, когда `dragging=false`.

Практика: Измените обработчик события:

```
onClipEvent (mouseMove) {
    if ( ! dragging ) {
        if (_root._xmouse < _root._width/2)
            _rotation -= 0.5;
        else _rotation += 0.5;
    }
}
```

Восклицательный знак означает операцию НЕ, так что условный оператор `if(!dragging)` имеет смысл «если не тащим ракету». Только при этом условии изменяется свойство `_rotation`.

Отличие кнопки от клипа

Как мы видели, кнопки и клипы могут обрабатывать события мыши с помощью обработчиков `on (...)`. Однако код кнопки не может содержать обработчиков `onClipEvent (...)`, которые предназначены только для клипов.

Важно: Главное отличие кнопки от клипа состоит в том, что кнопка (символ типа **Button**) — часть монтажного стола, на котором она находится, а клип (**Movie Clip**) имеет свой собственный монтажный стол.

Это означает, что у кнопок нет методов `play()` и `stop()`.

Когда мы используем команду `stop()`; или `this.stop()`; в обработчике события **кнопки**, мы останавливаем проигрывание на ее монтажном столе (в данном случае — на `_root`).

Если же эта команда используется внутри обработчика **клипа**, она останавливает проигрывание этого клипа (внутреннюю анимацию).

Если команда


```
_rotation += 45;
```

стоит в обработчике **кнопки**, на 45 градусов повернется ее монтажный стол, то есть `_root`. Та же самая команда в обработчике клипа повернет на 45 градусов этот клип.

А если нужно повернуть кнопку? Это тоже можно сделать, но иначе, используя ее относительный или абсолютный адрес. На панели **Properties** введем имя кнопки. Пусть, например, кнопка с именем `btn` находится на главном монтажном столе `_root`. Тогда ее адрес может быть записан как `_root.btn` или (при обращении с `_root`) просто `btn`. Повернуть кнопку в ее же обработчике можно с помощью команды

```
btn._rotation += 45;
```

или (из любого места программы!)

```
_root.btn._rotation += 45;
```

7.12 Клавиатура

Проще всего «ловить» нажатие клавиш с помощью события `keyPress`. В заголовке обработчика после слова `keyPress` в кавычках ставится нужный символ, например:

```
on (keyPress "A") { ... }
```

Для специальных клавиш указывают название в угловых скобках, например, "`<Left>`" (влево), "`<Right>`" (вправо), "`<Up>`" (вверх), "`<Down>`" (вниз).

Практика: В код клипа добавьте обработчики событий клавиатуры. Для клавиши «влево»:

```
on (keyPress "<Left>"){  
  _x -= 2;  
}
```

и аналогично для остальных клавиш-стрелок.

К сожалению, в одном обработчике можно задать только реакцию на одну клавишу.

Несколько большую свободу дает использование событий клипа `keyDown` и `keyUp`, но их обсуждение мы отложим до следующих уроков.

7.13. Практикум

Задание для работы: Для перехода к основной сцене нужно ввести **пароль** «123», щелкнуть мышкой в углублении скульптуры справа от слова *пароль* и ввести **пароль** «123». Создать просмотр побережья озера через иллюминатор. Кнопки позволяют перемещать фотографию на большие расстояния (50 пикселей по оси **X** и 20 пикселей по оси **Y**), а клавиши стрелки обеспечивают точную настройку (перемещение с шагом 5 пикселей). Здесь работает также и колесико мыши, с помощью которого можно смещать панораму влево и вправо.

Просмотр панорамы

В следующем примере мы применим обработчики событий мыши и клавиатуры, а также посмотрим, как защитить данные с помощью пароля.

Для перехода к основной сцене нужно щелкнуть мышкой в углублении скульптуры справа от слова *пароль* и ввести **пароль** «123».

Теперь мы смотрим на побережье озера через иллюминатор. Кнопки позволяют перемещать фотографию на большие расстояния (50 пикселей по оси **X** и 20 пикселей по оси **Y**), а клавиши стрелки обеспечивают точную настройку (перемещение с шагом 5 пикселей). Здесь работает также и колесико мыши, с помощью которого можно смещать панораму влево и вправо.

Несложно сообразить, что часть большого рисунка, которая не попадает в область видимости, закрыта с помощью слоя-маски.


Практика: Откройте файл `PRACTICE\7\panoram fla`. Переименуйте слой *Layer 1* в *Фон*.


Установите масштаб 50% (окно выбора масштаба в правом верхнем углу сцены) и перетащите на сцену клип *Панорама* из библиотеки.






На панели **Properties** дайте клипу имя `panoram` и установите его координаты `X=-725` и `Y=0`.

Изображение имеет размер 2000 на 400 пикселей, а сцена — 550 на 400 пикселей. Таким образом, центр рисунка совпадает с центром окна.

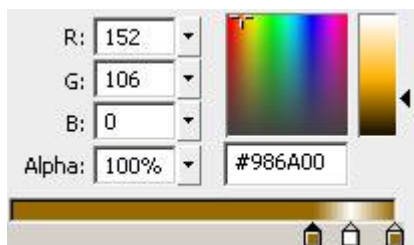
Практика: Отключите видимость слоя *Фон* и увеличьте масштаб до 100%. Создайте новый слой *Окно*, включите на панели инструментов режим рисования объектов (кнопка ) и нарисуйте черный прямоугольник без контура, занимающий всю сцену.

Практика: Выберите инструмент **Oval Primitive**  (под той же кнопкой, что и прямоугольник) и установите серый цвет заливки. нарисуйте круг, удерживая клавишу **Shift**. На панели **Properties** установите его размеры 300 на 300 пикселей, свойство **Inner radius = 80**.

Таким образом, мы нарисовали кольцо диаметром 300 пикселей, внутренний диаметр составляет 80% от внешнего, то есть 240 пикселей.

Практика: Выделите кольцо и выровняйте его по центру сцены с помощью панели **Align** (клавиши **Ctrl+K**, кнопки  и  при включенном режиме  — выравнивание относительно сцены).

Практика: С помощью панели **Color** установите для заливки кольца радиальный градиент, который придает объемный вид:



Следующий этап — создание маски, через которую видна панорама. Вспомните, что мы уже применяли слой-маски для создания сложных переходов между слайдами в **теме 5**.

Практика : Перетащите слой *Фон* выше слоя *Окно*. Создайте новый (самый верхний) слой *Маска* и нарисуйте на нем круг диаметром 240 пикселей (этот диаметр совпадает с диаметром внутренней части кольца). Выровняйте круг по центру сцены.

Практика: Нажмите правую кнопку мыши на названии слоя *Маска* и выберите команду **Mask** из контекстного меню. Сделайте видимым слой *Фон* и убедитесь, что теперь картинка видна через иллюминатор.

Теперь напишем код, с помощью которого будем смещать видимую часть рисунка. Фактически для этого нужно только изменять координаты `_x` и `_y` клипа `panoram`.

Практика: Создайте новый (самый верхний) слой *Кнопки* и добавьте на сцену кнопку из библиотеки. Уменьшите ее размеры и скопируйте еще 3 раза (перетащив при нажатой клавише **Alt**). Установите кнопки в нужные места и разверните каждую из них в правильном направлении. Для выравнивания кнопок на сцене можно использовать панель **Align**.

Добавим код к кнопкам, которые будут выполнять большие перемещения видимой области.

Практика: Добавьте к кнопке «влево» код обработчика

```
on (release) {  
    panoram._x += 50;  
}
```

Аналогичный код добавьте к остальным кнопкам. По оси **X** сделайте перемещение за 1 раз на 50 пикселей, а по оси **Y** — на 20. Проверьте работу кнопок.

Теперь добавим обработчики клавиш-стрелок, которые будут смещать рисунок на 5 пикселей (точная настройка). Эти обработчики можно присоединить к любой кнопке, поскольку все они получают событие `keyPress`.

Практика : Выделите любую кнопку и добавьте к ней обработчик клавиши «влево»:

```
on (keyPress "<Left>") {
  panoram._x += 5;
}
```

Аналогично здесь же запишите обработчики для остальных клавиш-стрелок. Проверьте работу клипа.

Обратите внимание, что при постоянном нажатии на клавиши рисунок непрерывно прокручивается.

Остался еще один недостаток — программа позволяет перемещать рисунок больше, чем нужно, открывая фон за рисунком. Чтобы исправить положение, мы напишем функцию `check`, которая будет проверять допустимость координат `_x` и `_y` и возвращать рисунок в разрешенное состояние. Эту функцию можно связать с первым кадром слоя *Кнопки*.

Вспомним, что размер рисунка — 2000 на 400 пикселей, размер сцены — 550 на 400 пикселей, а диаметр круга-маски — 240 пикселей. Несложные расчеты показывают, что `x`-координата левого верхнего угла рисунка должна находиться в интервале от -1605 до 155, а его `y`-координата — в интервале от -80 до 80.

Практика : Выделите первый кадр слоя *Кнопки* и добавьте к нему код

```
function check() {
  if ( panoram._x < -1605 ) panoram._x = -1605;
  if ( panoram._x > 155 ) panoram._x = 155;
  if ( panoram._y < -80 ) panoram._y = -80;
  if ( panoram._y > 80 ) panoram._y = 80;
}
```

Теперь в конец каждого обработчика (для всех кнопок и клавиш) нужно добавить вызов функции `check`, например, так:

```
on (release){
  panoram._x -= 50;
  check();
}
```

Сделайте это, сохраните файл и проверьте его работу.

Слушатели. Работа с колесиком мыши

В современных программах желательно обеспечить возможность разнообразного управления, в том числе — от колесика мышки. Оказывается, для этого достаточно добавить всего несколько строк кода.

Практика: Добавьте к кадру 1 слоя *Кнопки* такой код:

```
ear = new Object();
ear.onMouseWheel = function ( step ) {
  panoram._x += 5*step;
  check();
}
Mouse.addListener ( ear );
```

Запустите ролик, щелчком мыши сделайте его окно активным и проверьте, как работает горизонтальная прокрутка панорамы с помощью колеса мыши.

Теперь обсудим код. В строчке `ear = new Object();`

в памяти строится новый объект с именем `ear`. Что это за объект — пока неясно, мы не определили ни его свойства, ни методы.

Далее определяется метод `onMouseWheel` (*mouse* — мышь, *wheel* — колесо). Это значит, что объект может реагировать на событие `mouseWheel` (прокрутка колеса мыши), если об этом ему кто-нибудь сообщит. Обработчик события — это функция, принимающая один параметр — число шагов прокрутки. Как и насколько переместить панораму — определяем мы в теле функции (можно было, например, сделать вертикальную прокрутку). После изменения координат идет проверка с помощью функции `check`.

Самая важная строчка:

```
Mouse.addListener ( ear );
```

Здесь сказано, что объект `ear` становится «слушателем» (*listener*) глобального объекта `Mouse` (мышь), то есть, будет получать информацию о всех событиях, происходящих с мышью. В частности, при прокрутке колесика происходит событие `mouseWheel` и будет вызвана написанная нами функция.

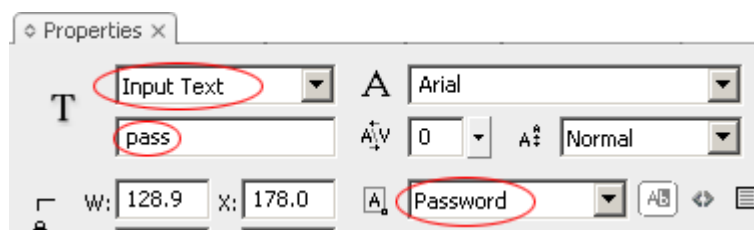
Защита паролем

Практика: Выделите кадр 1 во всех слоях и перетащите его мышкой на кадр 2.

В результате появился пустой кадр 1, в котором мы сделаем ввод пароля.

Практика: Выделите кадр 1 слоя *Окно* и перетащите на него рисунок `sphinx.jpg` из библиотеки. С помощью панели **Properties** установите координаты рисунка $X=0$ и $Y=0$.

Практика: Включите инструмент **T**. Установите шрифт **Arial** светлого цвета, размер 20. Добавьте новое текстовое поле в углублении под лапами сфинкса. Выберите тип поля: **Input text** (поле для ввода текста), стиль **Password** (ввод пароля). Дайте полю имя `pass`, по которому мы будем обращаться к нему из программы (см. рисунок ниже).



Стиль текста **Password** говорит о том, что вместо вводимых символов на экране видны только звездочки, чтобы посторонние не смогли увидеть верный пароль.

Теперь мы сделаем, чтобы при вводе пароля «123» ролик переходил на следующий слайд с просмотром панорамы.

Практика: Выделите кадр 1 слоя *Окно* и добавьте код:

```
stop();
pass.onChanged = function () {
  if ( pass.text == "123" ) nextFrame();
}
```

В первой строчке мы остановили проигрывание, чтобы не было автоматического перехода на второй кадр. Дальше идет новый прием — текстовому полю `pass` назначается обработчик события `changed` — оно возникает каждый раз, когда текст в поле изменяется. При этом будет вызываться написанная нами функция-обработчик.

В условном операторе содержимое текстового поля, которое имеет адрес `pass.text`, сравнивается с правильным паролем «123». Если получено совпадение, происходит переход к следующему кадру по команде `nextFrame`.

Практика: Добавьте слева от поля ввода надпись *пароль* (статический текст, тип **Static Text**). Проверьте работу фильма и сохраните его.

Вопросы для самостоятельной работы.

1. Как называется программа на *ActionScript*?
2. С какими элементами фильма может быть связан программный код?
3. Какие три простых типа переменных вы знаете?
4. Назовите **приоритет** (старшинство) арифметических операций в *ActionScript*
5. Объясните *Up, Over, Down* состояния кнопки.

Термины: переменные, объектно-ориентированное программирование, массив, условные операторы, циклы, функции

Литература:[[1](#) С.222-267 ; [2](#) С.97-236]

Тема 8. Объекты среды *Flash*

8.1. *Key* — клавиатура

Задание для работы: Создать клип «Ракета». Рисуем звездное небо и поверхность планеты. Создаем объект «Ракета», которым можно управлять клавишами «стрелками» вверх, вниз, вправо, влево, а также перемещать мышью. Ракета должна самостоятельно опускаться на поверхность планеты при отсутствии нажатия клавиш вверх и вниз. При нажатии клавиши **Ctrl** скорость движения увеличивается. Также нужно, чтобы Ракета не могла улететь вниз за пределы экрана.

Серьезное программирование во *Flash* строится на использовании понятия **объект**. Объект — это нечто, имеющее свойства («знания») и методы («умения»). Свойства — это данные, связанные с объектом, а методы — связанные с ним функции.

Для получения данных от клавиатуры удобно использовать глобальный объект *Key*. Чаще всего используются два его метода

- *Key.getCode()* — код нажатой клавиши (целое число);
- *Key.isDown(код символа)* определить, нажата ли клавиша с данным кодом; это логическая функция, она возвращает ответ *true* («да») или *false* («нет»).

Код символа для метода *isDown* найти по специальной таблице (например, на [сайте Adobe](#)).

Кроме того, самый простой способ — определить код экспериментально, используя функцию *getCode()* в обработчике *keyDown*.

Коды некоторых специальных клавиш можно получить как свойства объекта *Key*. В первом примере мы будем при каждом нажатии клавиши выводить в окно **Output** ее код и состояния клавиш **Ctrl, Shift** и **Alt** (*true* — нажата, *false* — не нажата).

Изучим эти возможности на примере проекта «Ракета2». Он управляется клавишами-стрелками и перетаскивается мышкой. При нажатой клавише **Ctrl** скорость движения увеличивается.

Практика: Создаем новый слой Фон. Рисуем звездное небо и поверхность планеты. Создаем графический объект «Ракета1».

Пример:



Создаем объект *Movie Clip* «Ракета» и вставим наш объект «Ракета1».

Создаем новый слой Ракета и вставляем в него объект *Movie Clip* «Ракета».

Практика: Добавьте код, связанный с ракетой:

```
onClipEvent (keyDown) {  
    trace(Key.getCode());  
    trace(Key.isDown(Key.CONTROL));  
    trace(Key.isDown(Key.SHIFT));  
    trace(Key.isDown(Key.ALT));  
}
```

Проверьте работу клипа, нажимая на разные клавиши.

Теперь посмотрим, как можно управлять ракетой с клавиатуры. Событие `keyDown` происходит только при нажатии клавиши, а нам нужно, чтобы ракета двигалась постоянно, если мы нажали клавишу и не отпускаем ее. Для этого мы используем событие `enterFrame`, где с помощью метода `Key.isDown` будем проверять, какие клавиши нажаты, и менять соответствующим образом координаты `_x` и `_y` клипа.

Скорость движения запишем в переменную `v`, ее начальное значение зададим в обработчике `load` (за время одного кадра ракета смещается на 3 пикселя в заданном направлении).

Практика: Добавьте новые обработчики событий для ракеты:

```
onClipEvent (load) {  
    v = 3;  
}  
onClipEvent (enterFrame) {  
    if (Key.isDown(Key.RIGHT)) {  
        _x += v;  
    } else if (Key.isDown(Key.LEFT)) {  
        _x -= v;  
    }  
}
```

Аналогично запишите обработчики для клавиш-стрелок «вверх» и «вниз» (коды `Key.UP` и `Key.DOWN`).

Заметьте, что ракета может двигаться одновременно в двух направлениях, например, вверх и влево.

Теперь сделаем так, чтобы при нажатии клавиши **Home** ракета возвращалась в исходное состояние (где она была в начале). Для этого в обработчике события `load` надо запомнить ее координаты в переменных `x0` и `y0`, а при нажатии клавиши **Home** присваивать эти значения координатам клипа `_x` и `_y`.

Практика: Добавьте строчки в обработчик `load`:

```
x0 = _x;  
y0 = _y;
```

Измените обработчик события `keyDown` на такой:

```
onClipEvent (keyDown) {  
    if (Key.isDown(Key.HOME)) {  
        _x = x0;  
        _y = y0;  
    }  
}
```

Запустив ролик, вы можете с удивлением обнаружить, что это не работает. Если вставить строчку

```
trace("Ку-ку");
```

в обработчик `keyDown`, мы обнаружим, что при нажатии клавиши **Home** этот обработчик не вызывается.

Дело в том, что мы запускаем клип из среды разработки, которая перехватывает некоторые клавиши, например, **Enter** и **Home**. При просмотре этого ролика в отдельном окне *Flash*-проигрывателя или в браузере все будет нормально.

Однако есть простой способ избавиться от этого неудобства.

Практика: Запустите ролик и отметьте флажок **Disable Keyboard Shortcuts** в меню **Control** в окне проигрывателя. Проверьте, реагирует ли теперь клип на нажатие клавиши **Home**.

Теперь сделаем так, чтобы при нажатии клавиши **Ctrl** скорость движения ракеты увеличивалась в 3 раза. Начальную скорость мы обозначим через `v0`.

Практика: Добавьте в начало обработчика события `enterFrame` код

```
if (Key.isDown(Key.CONTROL)) {  
    v = v0*3;  
} else {  
    v = v0;  
}
```

и измените первую строчку в обработчике `load` на

```
v0 = 3;
```

Практика: Самостоятельно сделайте так, чтобы ракета опускалась вниз, если не нажата ни стрелка «вверх», ни стрелка «вниз». Также нужно, чтобы она не могла улететь вниз за пределы экрана.

Далее мы сделаем так, чтобы ракету можно было перетаскивать по экрану мышью. В теме 7.11 мы рассматривали код для перемещения объекта мышью. Используйте его.

Практика: Подсказка

```
//перемещение объекта мышью  
on (press) {  
    startDrag(this);  
}  
on (release, releaseOutside) {  
    stopDrag();  
}
```

8.2. Работа с текстом (*String*, *Selection*)

Задание для работы: Построить небольшой редактор текста, который умеет делать выделенные буквы заглавными или строчными, считать количество символов и искать в тексте заданное слово.

Объект *String* — строка

Символьные строки в программе заключаются в двойные кавычки и представляют собой объекты класса **String** (строка). У строки есть одно свойство

- `length` — длина строки,

и несколько методов, из которых чаще всего применяются

- `charAt(n)` — символ в позиции `n` (номера символов начинаются с нуля);
- `charCodeAt(n)` — код символа в позиции `n` (целое число);
- `substring(from, to)` — подстрока, которая начинается с позиции `from` и заканчивается **перед** позицией `to` (то есть, символ с номером `to` в нее не входит); если второй аргумент не указан, выделяется фрагмент до конца строки;
- `indexOf(sub)` — ищет подстроку `sub`, возвращает номер ее первого символа; если такой подстроки нет, возвращает `-1`;
- `toLowerCase()` — преобразовать в строчные (маленькие) буквы;
- `toUpperCase()` — преобразовать в заглавные (большие) буквы.

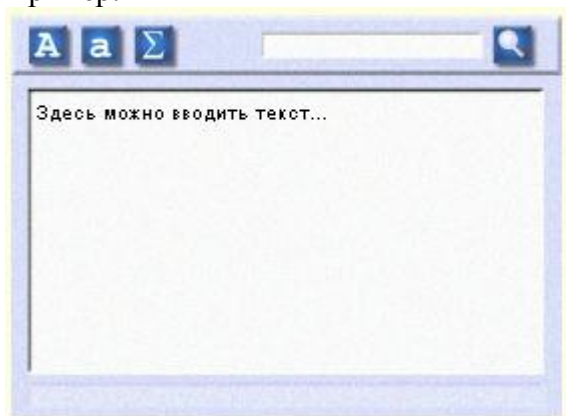
Для того, чтобы соединять строки, используется знак `+`, например, так:

```
s = "123";  
t = s + 45 + "6789";  
trace ( t ); // получим "123456789"
```

Во второй строке число **45** было автоматически преобразовано к символьной форме.

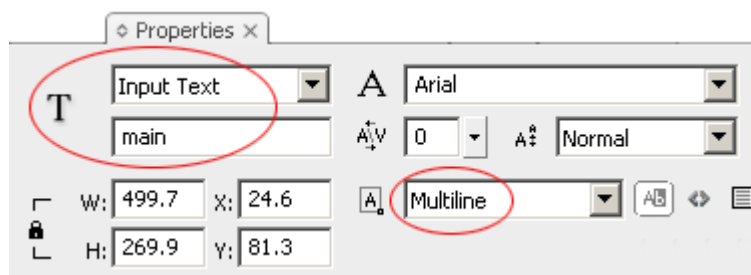
В следующем проекте мы построим небольшой редактор текста, который умеет делать выделенные буквы заглавными или строчными, считать количество символов и искать в тексте заданное слово.

Пример:



Практика: Создаем новый проект, рисуем фон для текстового редактора. Создаем четыре кнопки **A**, **a**, **Σ**, **🔍**. Разместить кнопки как в примере.

Практика: Создайте новый слой *Текст* и разместите большое текстовое поле в середине сцены. Установите тип надписи **Input Text** (поле ввода), имя **main** и параметр **Multiline** (многострочный текст), как показано на рисунке ниже.



Содержимое текстового поля — это тоже строка, то есть, объект класса *String*. Она имеет имя `text`, это значит, что адрес текстовой строки поля `main` запишется как `main.text`. Сделаем так, чтобы при нажатии кнопок **A** и **a** все буквы текста становились заглавными (или соответственно строчными).

Практика : Добавьте к кнопке **A** код

```
on ( release ){
    main.text = main.text.toUpperCase();
}
```

а к кнопке **a** — код

```
on ( release ){
    main.text = main.text.toLowerCase();
}
```

Проверьте работу кнопок.

В нижней части поля сделаем информационную строку (динамическое текстовое поле) и при нажатии на кнопку **Σ** будем выводить там количество символом в тексте, то есть длину строки `main.text`.

Практика: Добавьте в нижнюю часть сцены однострочный динамический текст (**Dynamic Text, Single line**) с именем **info**. Для кнопки **Σ** введите обработчик события

```
on ( release ){
    info.text = "Символов: " + main.text.length;
}
```


Объект *Selection*


Объект **Selection** — это выделенная часть текста в одном из полей ввода. Он связан с тем элементом, который в данный момент имеет **фокус**, то есть принимает команды от клавиатуры.

Методы объекта **Selection**:

- `setFocus("адрес")` — передать фокус ввода текстовому полю, адрес которого указан в кавычках;
- `getBeginIndex()` — возвращает номер первого выделенного символа;
- `getEndIndex()` — возвращает номер символа, на котором выделение заканчивается (этот символ уже не входит в выделение);
- `setSelection(start,end)` — выделить область от символа с номером `start` до символа с номером `end`.

Добавим возможность поиска слова в тексте.

Практика : Добавьте в верхнюю часть сцены (слева от кнопки ) однострочное поле ввода (**Input Text, Single line**) с именем **find**.



Поиск начинается по щелчку на кнопке  .

Практика : Для кнопки  задайте обработчик события

```
on (release) {
    if ( find.text == "" ) return;
    n = main.text.indexOf ( find.text );
    if ( n < 0 )
        info.text = "Ничего не найдено.";
    else {
        info.text = "Нашли в позиции " + n;
        Selection.setFocus ( "main" );
        Selection.setSelection ( n, n + find.text.length );
    }
}
```

Сначала проверяем, введено ли что-то в поле для поиска (если оно пусто, ничего делать не надо, и происходит выход из обработчика).

Затем, с помощью метода `indexOf` ищем образец в тексте. Если такого слова нет, выводит сообщение об этом в информационную строку. Если есть, то устанавливаем фокус на поле `main` и выделяем найденное слово.

Используя методы объекта **Selection**, можно улучшить работу кнопок  и : если что-то выделено, они будут менять только выделенную часть текста.

Сначала мы определим начало и конец выделенной части и запишем номера этих символов в переменные `nStart` и `nEnd`. Если они равны, то ничего не выделено и преобразуется весь текст. Если эти значения не равны, новая строка строится так: содержимое `main.text` «разрезается» на 3 части с помощью метода `substring`, ко второй части применяется метод `toUpperCase`.

Важно: Эта часть программы будет работать только для *Flash Player* версии 9 и выше, поскольку в ранних версиях во время щелчка по кнопке текстовое поле теряет фокус.

Практика : Измените обработчик кнопки  на такой:

```
on (release) {
    nStart = Selection.getBeginIndex();
    nEnd = Selection.getEndIndex();
    if (nStart == nEnd) {
        main.text = main.text.toUpperCase();
    } else {
```

```

s = main.text;
s = s.substring ( 0, nStart ) +
    s.substring ( nStart, nEnd ).toUpperCase() +
    s.substring ( nEnd );
main.text = s;
}
}

```

Аналогично измените код кнопки **a** .

Обратите внимание, что при вызове `substring(nEnd)` второй аргумент не указан, то есть выделяется фрагмент с символа `nEnd` до конца строки.

Заметим, что алгоритм работает даже тогда, когда поле `main` не имеет фокуса ввода (и обе функции `getBeginIndex` и `getEndIndex` вернут -1).

8.4. Color — цвет

Задание для работы: Нарисуйте знак доллара, превратите его в объект `Movie Clip` с именем **logo**. Примените к нему фильтры `Bevel` и `Glow`. Создайте кнопку и разместите внизу сцены девять таких кнопок. Написать программу для управления цветом объекта с помощью кнопок в нижней части сцены.

Объект `Color` позволяет изменять цвет клипов из программы. Для этого нужно сначала создать в памяти новый объект, связав его с клипом:

```
col = new Color ( адрес клипа );
```

Таким образом, мы получаем доступ к цвету объекта. Затем можно использовать два метода:

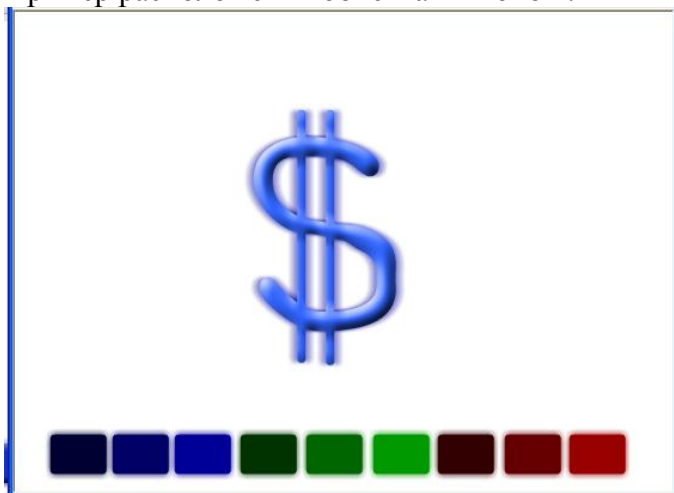
- `setRGB(цвет)` — установить новый цвет;
- `getRGB()` — получить цвет, установленный последней командой `setRGB`.

Цвет клипов задается в формате `RGB` (`R=red`, красный; `G=green`, зеленый; `B=blue`, синий) как длинное целое число, в котором «упакованы» значения составляющих **R**, **G** и **B**, каждая из которых представляет собой целое число от 0 до 255.

Чтобы удобно было разбираться в цветах, принято записывать их в шестнадцатеричной системе счисления, так что младшие две цифры — это **B**, следующие две — **G**, а две старших — **R**. Например, для цвет `0x660000` имеем **R=66₁₆=102**, **G=B=0**.

В проекте этого раздела мы будем управлять цветом объекта с помощью кнопок в нижней части сцены.

Пример расположения объекта и кнопок :



Практика: Нарисуйте знак доллара, превратите его в объект `Movie Clip` с именем **logo**. Примените к нему фильтры `Bevel` и `Glow`. Создайте кнопку и разместите внизу сцены девять таких кнопок.

Практика : Выделите самую левую кнопку и перейдите на панель **Properties**. В списке **Color** выберите вариант **Tint** (цветовой оттенок), цвет с кодом **#000033** и смещение 100% (полная замена исходного цвета).

Практика : Выделите первую кнопку и введите код обработчика, который вызывается при отпускании кнопки мыши:

```
on ( release ) {  
    cLogo = new Color(_root.logo);  
    cLogo.setRGB(0x000033);  
}
```

Аналогично задайте параметры и обработчик второй кнопки с цветом **0x000066**. Проверьте работу фильма.

Конечно, можно применить те же шаги для остальных кнопок, но эту нудную работу можно значительно упростить, если использовать массивы и циклы.

Массивы

Массив — это набор элементов, имеющих общее имя. Каждый элемент имеет собственный номер, начиная с **нуля**. Для обращения к элементу массива используются квадратные скобки.

Важно: Нумерация элементов массива с **нуля** непривычна для обычных людей, но широко используется в программировании (языки *Cu*, *JavaScript*, *Java*, *ActionScript*).

В отличие от многих языков программирования, в одном массиве могут быть разнотипные элементы: целые числа, вещественные числа, строки, логические переменные.

Массив из трех элементов можно объявить и заполнить так:

```
A = new Array(3);
```

```
A[0] = 12;
```

```
A[1] = 4.56;
```

```
A[2] = "Ку-ку!";
```

или так:

```
A = new Array (12, 4.56, "Ку-ку!");
```

или так:

```
A = [12, 4.56, "Ку-ку!"];
```

При работе с массивами, как и с другими объектами (кроме чисел, строк и логических переменных), есть один тонкий момент. Рассмотрим пример кода:

```
A = [1, 2, 3];
```

```
B = A;
```

```
B[1] = 99;
```

```
trace ( A );
```

Здесь создается массив **A**, затем он копируется в массив **B**. Потом изменяется **B[1]** и массив **A** выводится в окно **Output**. Вроде бы массив **A** не должен измениться, однако мы увидим в окне **Output** строчку:

```
1,99,3
```

Почему же изменился массив **A**? Дело в том, что оператор **B=A** НЕ создает новый массив, а просто копирует в **B** адрес массива **A**, то есть **A** и **B** обращаются к одной области памяти.

Поэтому, изменив **B**, мы изменили и **A**. Чтобы действительно создать копию массива, нужно заменить оператор **B=A** на код

```
B = new Array();
```

```
for(i=0; i<A.length; i++) B[i] = A[i];
```

Здесь используется встроенное свойство **length** объекта **Array** — длина массива.

В нашем примере мы создадим массив, в котором будут храниться коды цветов всех кнопок.

Практика : Удалите ранее созданные обработчики. Создайте новый слой *Программа* и в первый кадр введите код, заполняющий массив кодами цветов:

```
colors = [0x000033, 0x000066, 0x000099,  
          0x003300, 0x006600, 0x009900,  
          0x330000, 0x660000, 0x990000];
```

Обратим внимание, что обработчики события `release` для всех кнопок почти одинаковые и отличаются только кодом цвета. Поэтому можно создать для них одну единственную функцию, которую будут вызывать все кнопки.

Важно : Слой *Программа* можно сразу же заблокировать, щелкнув по значку в столбце . Это позволит добавлять код *ActionScript*, но не даст случайно добавить что-то еще.

Практика : Добавьте функцию, которая изменяет цвет логотипа в соответствии с цветом кнопки, на которой щелкнули мышкой:

```
function changeColor() {
    c = new Color(this);
    col = c.getRGB();
    cLogo = new Color(_root.logo);
    cLogo.setRGB(col);
}
```

В этой функции мы сначала определяем цвет кнопки, получив доступ к нему через переменную `c` и используя метод `getRGB`. Затем этот цвет переносится на логотип. Созданная функция пока не работает, поскольку мы не определили, в каких случаях она вызывается. Для того, чтобы она стала обработчиком события `release` некоторой кнопки, достаточно написать

`Кнопка.onRelease = changeColor;`

Далее мы с помощью цикла, в котором переменная `i` (номер кнопки) меняется от 0 до 8, настроим все кнопки сразу: перекрасим их в нужный цвет и добавим обработчик события.

Чтобы в цикле обращаться к кнопке по ее номеру, можно использовать специальную форму `_root[имя]`. Таким образом, `_root["color0"]` это то же самое, что и `_root.color0`, но преимущество первого метода в том, что имя кнопки можно построить динамически, во время выполнения, например, так:

```
_root["color"+chr(48+i)]
```

Функция `chr` преобразует код символа в символ. Учитывая, что цифры **0-9** имеют коды **48-57**, при `i=2` получаем `_root["color2"]`.

Чтобы эта идея сработала, кнопкам нужно дать соответствующие имена **color0**, **color1** и т.д.

Практика : Дайте имена кнопкам **color0**, **color1** и т.д., начиная с левой. На панели **Properties**, выделив какую-нибудь кнопку.

Практика : Добавьте к коду кадра 1 цикл:

```
for (i=0; i<9; i++){
    obj = _root["color"+chr(48+i)];
    c = new Color(obj);
    c.setRGB(colors[i]);
    obj.onRelease = changeColor;
}
```

Проверьте работу фильма и сохраните его.

Здесь переменная `obj` — это адрес кнопки с номером `i`.

8.4. *Sound* — звук

Объекты класса *Sound* служат для управления звуком. С их помощью можно

- изменять **громкость** звука;
- изменять распределение звука между левым и правым каналами, когда источник звука перемещается по экрану;
- создавать звуки динамически, т.е. из программы без использования временной шкалы.

Сначала научимся проигрывать звук из библиотеки и останавливать его в нужный момент.

Практика: Найдите звук в библиотеке и нажмите на нем правую кнопку мыши. В контекстном меню выберите команду **Linkage...** (связывание), в появившемся окне отметьте флажки **Export for ActionScript** и **Export in first frame**, введите имя звука например - **hey** в поле **Identifier**.

Флажок **Export for ActionScript** делает звук доступным из программы, мы будем обращаться к нему по имени **hey**. Флажок **Export in first frame** загружает звук вместе с первым кадром, таким образом, звук будет доступен с самого начала фильма.

Теперь надо

- создать новый объект типа Sound с помощью конструктора `new Sound()`;
- загрузить в него звук с помощью метода `attachSound`;
- запустить звук на проигрывание (метод `start`)
- остановит звук с помощью метода `stop`.

Мы свяжем этот код с клипом, поскольку он будет получать события `mouseDown` и `keyDown` вне зависимости от того, где находится мышка. Объект Sound строится сразу после загрузки в обработчике `load`, звук запускается в обработчике `mouseDown` и останавливается в случае события `keyDown`.

Практика: Выделите клип и добавьте в окно **Actions** следующий код:

```
onClipEvent (load) {
    snd = new Sound();
}
onClipEvent (mouseDown) {
    snd.attachSound ("hey");
    snd.start(0, 1);
}
onClipEvent (keyDown) {
    snd.stop();
}
```

При вызове метода `start` первое число обозначает время от начала звукового файла в секундах (0 — с самого начала), а второй — количество повторов.

Важно: Если проигрывать звук не с начала, после завершения звуковой дорожки будет проиграна начальная часть.

Важно: С помощью вызова `stopAllSounds()`; можно прекратить проигрывание всех звуков сразу.

Чтобы «добраться» до звука внутри клипа, нужно указать адрес клипа при создании объекта Sound. Например, в коде самого клипа можно использовать `this`:

```
bounce = new Sound(this);
```

Для изменения громкости используется метод `setVolume`, в скобках нужно указать громкость звука в процентах от исходной.

Звуковой баланс регулируется с помощью метода `setPan` (*set panoram* — установить баланс), в скобках нужно задать величину от -100 (левая граница зоны, только левый канал) до 100 (правая граница, только правый канал).

8.5 Дата и время

Задание для работы : Построить ролик, который показывает сегодняшнюю дату, день недели и время, а также может служить будильником.

Объект *Date*

В этом разделе мы изучим средства для работы с датами и временем:

- объект `Date` (дата);
- функцию `getTimer()` (получить время таймера).

Создать новый объект класса `Date` можно так:

```
d = new Date(2013, 4, 09);
```

Первое число — год, второе — номер месяца, третье — число. Эта дата — 09 мая 2013 года. Здесь нет ошибки, номера месяцев начинаются с нуля, поэтому месяц 4 в программе *Flash* — это май.

Если параметры не используются:

```
today = new Date();
```

создается объект с сегодняшней датой.

С помощью методов объекта *Date* можно определить год, месяц, число и день недели:

- `today.getFullYear()` — год;
- `today.getMonth()` — месяц;
- `today.getDate()` — число;
- `today.getDay()` — день недели;
- `today.getHours()` — часы;
- `today.getMinutes()` — минуты;
- `today.getSeconds()` — секунды.

Дни недели, так же, как и месяцы, нумеруются с нуля, причем неделя начинается с воскресенья (день 0), понедельник имеет номер 1 и т.д.

Пример :



Практика : Нарисуйте фон и стилизацию отрывного календаря.

Практика : Добавьте динамические текстовые поля (**Dynamic Text**) для вывода дня недели, числа, месяца года так, как на образце. Дайте им имена `day`, `date`, `month` и `year`, установите выравнивание по центру, выберите цвет и размер шрифта.

Практика: Добавьте слой *Программа* и в первом кадре введите код `stop();`

```
today = new Date();
```

```
weekDays = ["воскресенье", "понедельник", "вторник",  
            "среда", "четверг", "пятница", "суббота"];
```

```
months = ["января", "февраля", "марта", "апреля",  
          "мая", "июня", "июля", "августа", "сентября",  
          "октября", "ноября", "декабря"];
```

```
day.text = weekdays[today.getDay()];
date.text = today.getDate();
month.text = months[today.getMonth()];
year.text = today.getFullYear();
```

Проверьте работу фильма.

В первой строчке мы остановили проигрывание, поскольку все изменения выполняются из программы. Затем в массивы `weekDays` и `months` записаны названия дней недели и месяцев (учитывая, что неделя у американцев начинается с воскресенья).

Затем в текстовые поля записываются данные сегодняшнего дня, причем некоторые из них выбираются из массивов.

С выводом времени дело обстоит несколько сложнее, его надо обновлять раз в секунду. Сначала добавим текстовое поле и напишем функцию, которая должна выводить в него время.

Практика : Добавьте текстовое поле под листом календаря и дайте ему кодовое имя `time`. В код кадра 1 добавьте функцию

```
function showTime() {
  function str2 ( n ) {
    var s = String(n);
    if ( n < 10 ) s = "0" + s;
    return s;
  }
  today = new Date();
  h = today.getHours();
  m = today.getMinutes();
  s = today.getSeconds();
  time.text = str2(h) + ":" + str2(m) + ":" + str2(s);
}
```

Эта функция содержит внутреннюю функцию `str2`, которая принимает целое число `n` и переводит его в строку с помощью функции `String`. Затем, если число было меньше 10, к строке спереди добавляется ноль. Это сделано затем, чтобы, скажем, время 5 часов 3 минуты и 2 секунды изображались как `05:03:02`, а не `5:3:2`.

В основной части функции создается новый объект `Date`, содержащий текущую дату и время. С помощью методов `getHours`, `getMinutes` и `getSeconds` из него извлекаются часы, минуты и секунды. Далее они переводятся в символьный вид с помощью функции `str2` и объединенная строка записывается в свойство `text` поля `time`.

Все бы хорошо, но функцию `showTime` надо вызывать периодически с интервалом не менее 1 секунды. К счастью, в среде *Flash* есть такое средство.

Практика : Добавьте к коду кадра 1 строчку

```
setInterval ( showTime, 200 );
```

и проверьте работу клипа.

Первый параметр функции `setInterval` — это функция, которую надо вызывать, а второй — интервал между вызовами в миллисекундах.

Таймер

Займемся будильником.

Практика : Добавьте на поле надписи (**Static Text**): Будильник, *время, сек, повторить и раз*, а также два поля ввода (**Input Text**) с именами `alarmTime` (время сигнала) и `loops` (повторы). Создайте кнопку ПУСК. Добавьте в нужное место кнопку *Пуск*.

Практика : В кадр 1 слоя *Программа* добавьте присвоение начальных значений:

```
alarmTime.text = "10";  
loops.text = "1";
```

Для отслеживания времени будем использовать внутренний таймер. Вызов функции `t = getTimer();`

записывает в переменную `t` количество миллисекунд, прошедшее с начала проигрывания фильма. Для будильника нам нужна разница между текущим временем и временем начала отсчета, которое мы запомним в специальной переменной `startTime`.

В другой переменной, `alarmOn` будем запоминать состояний будильника (включен или выключен). Теперь можно написать код для кнопки: при ее нажатии включается будильник и запоминается время начала отсчета.

Практика : Добавьте код обработчика для кнопки:

```
on (release) {  
    clock.alarmOn = true;  
    clock.startTime = getTimer();  
}
```

Остальная работа выполняется клипом `clock`: вращение стрелки, тиканье раз в секунду и выдача сигнала будильника.

Практика: Создать объект `Movie Clip` с именем `clock`. Нарисовать белый круглый циферблат с указанием 0, 15, 30 и 45 секунд.

Практика: Создать объект `Movie Clip` с именем `hand`. Нарисовать стрелку секундомера.

Практика : Вставить стрелку в циферблат.

Практика: Разместить готовый секундомер на главной сцене.

Практика: Из темы 7 проект *ship* взять звук *click.wav*. Переименовать его в *tick*. Из темы 6 взять звук *kuranty.mp3*. Переименовать его в *alarm*.

Практика : В контекстном меню выберите команду **Linkage...** (связывание), в появившемся окне отметьте флажки **Export for ActionScript** и **Export in first frame**, введите имя звуков соответственно *tick* и *alarm* в поле **Identifier**.

Практика : Добавьте к клипу-секундомеру код обработчика `enterFrame`:

```
onClipEvent (enterFrame) {  
    if ( !alarmOn ) return;  
    t = Number(_root.alarmTime.text);  
    diff = Math.round((getTimer()-startTime)/1000);  
    if (hand._rotation != diff*6) {  
        hand._rotation = diff*6;  
        tick = new Sound();  
        tick.attachSound("tick");  
        tick.start(0,1);  
    }  
    if (diff == t) Alarm();  
}
```

Если будильник не включили, ничего делать не надо и функция заканчивает работу.

В переменную `t` записывается числовое значение установленного времени, для перевода из символьной строки в число используется функция `Number`.

Разница между текущим и стартовым временем переводится в секунды из миллисекунд (делением на 1000) и округляется до ближайшего целого числа с помощью функции `Math.round`, которая относится к объекту `Math` (математика).

Если вспомнить, что за 1 секунду стрелка должна повернуться на 6(=360/60) градусов, то нужный угол поворота стрелки равен `diff*6`. Если стрелку надо «довернуть», изменяем ее свойство `_rotation` и 1 раз проигрываем звук `tick` из библиотеки.

Если разница `diff` совпала с установленным временем `t`, вызывается функция `Alarm`, которую мы сейчас напишем. В ней надо отключить будильник, вернуть стрелку в исходное положение и проиграть звук `alarm` из библиотеки нужное число раз.

Практика : Введите код обработчика события `load` для клипа-секундомера:

```
onClipEvent (load) {  
  function Alarm() {  
    alarmOn = false;  
    hand._rotation = 0;  
    snd = new Sound();  
    snd.attachSound("alarm");  
    snd.start(0,_root.loops.text);  
  }  
}
```

Проверьте работу будильника.

Заметим, что эта функция расположена в обработчике `load`, то есть она доступна с момента загрузки клипа в память.

Периодический вызов функции

Теперь добавим надпись «Пора вставать!», которая должна исчезать через 2 секунды после начала сигнала.

Практика : Добавьте динамический текст «Пора вставать!» на сцену с часами и дайте ему имя `wakeup`. Выберите шрифт и примените фильтры на ваш вкус.

Сначала эту надпись надо скрыть.

Практика : Добавьте в код кадра 1 слоя *Программа* строчку `wakeup._visible = false;`

Когда звенит будильник, мы откроем эту надпись, изменив ее свойство `_visible` на `true`. Вспомним, что мы уже использовали средство, которое позволяет вызывать функцию периодически с заданным интервалом. Например, вращение некоторого клипа `qq` можно организовать так:

```
function rot() {  
  qq._rotation += 10;  
}
```

```
setInterval ( rot, 1000 );
```

Через 1000 мс (или через 1 сек) будет вызываться функция `rot`, так что каждую секунду клип будет поворачиваться на 10 градусов.

Ссылку на интервал можно запомнить в переменной:

```
i = setInterval ( rot, 1000 );
```

Тогда для прекращения этих вызовов нужно вызвать функцию `clearInterval`:

```
clearInterval ( i );
```

или просто удалить из памяти эту переменную:

```
delete i;
```

Теперь применим этот способ к нашему случаю.

Практика : Добавьте в конец функции `Alarm` строчки

```
_root.wakeup._visible = true;  
i = setInterval ( removeText, 2000 );
```

После этой функции (внутри обработчика события `load`) добавьте еще одну функцию:

```
function removeText() {  
  _root.wakeup._visible = false;  
  delete i;
```

}

Проверьте работу клипа и сохраните его.

Вопрос для самостоятельной работы.

1. Что означает свойство `length` объекта класса **String** ?
2. Что означают методы `setRGB(цвет)` и `getRGB()` ?
3. Что можно сделать со звуком с помощью объекта класса *Sound* ?
4. Как нумеруются дни недели и месяцы в *ActionScript* ?

Термины: переменные, объектно-ориентированное программирование, объект, массив, условные операторы, циклы, функции.

Литература: [[1](#) С.280-462 ; [2](#) С.100-320]