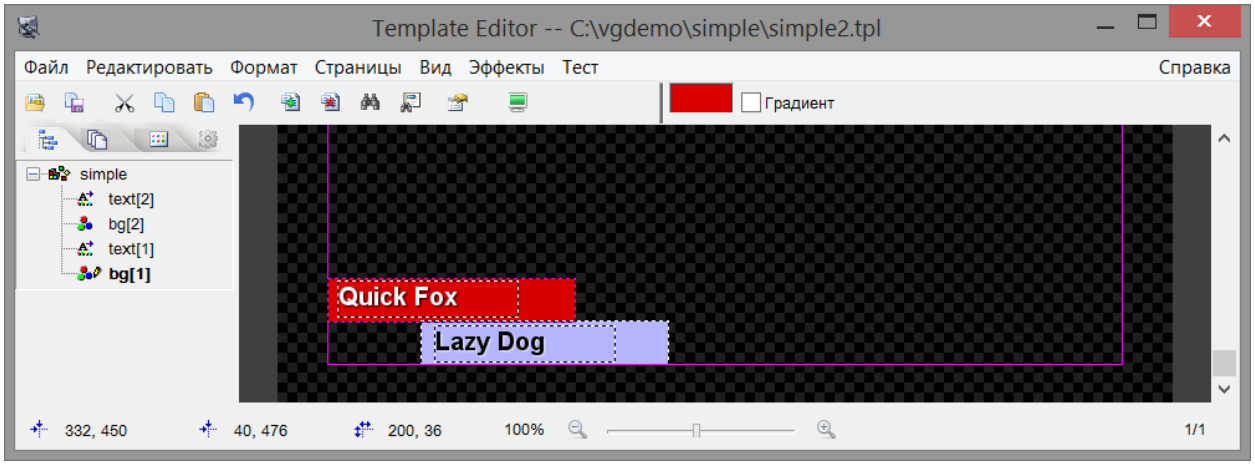
VGCAST - УЧЕБНИК

Простой шаблон без эффектов (simple1.tpl)

Создаем шаблон с несколькими слоями, слои **bg[1], bg[2]** -- подложка типа сплошной цвет, слои **text[1], text[2]** -- текстовые слои.



Если выполнить команду "**Эффекты**" - "**Загрузить шаблон ...**" то сформированный шаблон будет загружен в движок **vgcast** и сразу появится в эфирном канале. По умолчанию команде "**Загрузить шаблон ...**" назначен шорткат **F3**, поэтому можно просто давить **F3** на клавиатуре.

В таком режиме создается шаблон по умолчанию...

Атрибут шаблона "**Load Invisible**" (Загружать невидимым) **выключен**. Атрибут шаблона "**Global**" (Глобальный) тоже **выключен**. Атрибуты шаблона можно посмотреть в редакторе эффектов.

В этом случае шаблон загружается в движок как **видимый**, т.е. **все** его слои сразу выводятся в графическую память платы.

Шаблон загружается в движок командой меню "**Эффекты**" - "**Загрузить шаблон** ...". Этой команде по умолчанию назначен клавиатурный шорткат **F3**. Таким образом, если просто создать шаблон (не заморачиваясь с эффектами) то достаточно после создания шаблона нажать **F3** -- и шаблон в эфире.

Убрать шаблон из эфира можно либо командой меню "**Эффекты**" - "**Удалить шаблон ...**", по умолчанию этой команде назначен шорткат **F4**.

Атрибут **Global** управляет тем, как редактор загружает шаблоны в движок. Если этот атрибут выключен, то перед загрузкой шаблона сначала выполняется команда **удаления** шаблона с указанным именем. Этот атрибут нужен в тех случаях, когда ведется работа с многостраничным документом. Если на нескольких страницах документа используются шаблоны с одинаковыми именами, то при выводе страниц друг за другом, шаблоны с каждой страницы будут выводиться правильно, т.е. сначала будет удален шаблон уже загруженный в движок, и только потом будет загружен новый шаблон.

Что такое эффект?

Работа без эффектов -- это для совсем ленивых... Если хочется чтобы слои шаблонов появлялись "красиво" нужно использовать эффекты.

Эффект -- это последовательность команд движка vgcast.

В каждой команде задается ее смещение относительно начала эффекта.

Каждая команда должна быть расположена на отдельной строке.

Формат команды в редакторе эффектов

время **команда** ***параметры***

где **время** задает смещение команды относительно начала эффекта. Время может задаваться либо как количество кадров, либо в формате hh:mm:ss:ff, **команда** -- это имя команды движка.

Все элементы команды отделяются друг от друга одними или несколькими пробелами или табуляциями.

Для работы с эффектами атрибут шаблона "**Load Invisible**" (Загружать невидимым) должен быть **включен**. Атрибут шаблона "**Global**" (Глобальный) тоже должен быть **включен**. Когда включен режим "Загружать невидимым" все слои шаблона при загрузке в движок будут загружаться с прозрачностью 0, т.е. они будут невидимыми.

Эффекты по умолчанию (simple2.tpl)

При создании шаблона создается два эффекта по умолчанию -- эффект **show** (показать) и эффект **hide** (спрятать). Эффект **show** состоит из одной команды

00:00:00:00 fi @.*

Т.е. со смещением 0 относительно начала эффекта выполняется команда **fi** (fadein), изменить прозрачность слоя от 0 (полностью прозрачный) до 256 (полностью непрозрачный). В этой команде есть обязательный параметр -- точное имя слоя, т.е.

имя_шаблона.имя_слоя

В качестве имени шаблона можно использовать знак **@**, это имя текущего шаблона. В именах шаблонов и слоев можно использовать метасимволы ***** (любая строка) и **?**(любой знак). Таким образом, команда **fi @.*** применяется к ко всем слоям текущего шаблона и в результате выполнения команды каждый слой шаблона становится полностью непрозрачным, т.е. видимым.

Эффект **hide** состоит из двух команд

00:00:00:00 fo @.*
00:00:00:00 del @

Команда **fo** (fadeout) изменяет прозрачность слоя от 256 (полностью непрозрачный) до 0 (полностью прозрачный). Параметр команды указывает, что она применяется ко всем слоям текущего шаблона. Вторая команда удаляет шаблон (опять таки **@** = текущий шаблон) из движка. При удалении шаблона освобождаются все ресурсы, распределенные для слоев шаблона.

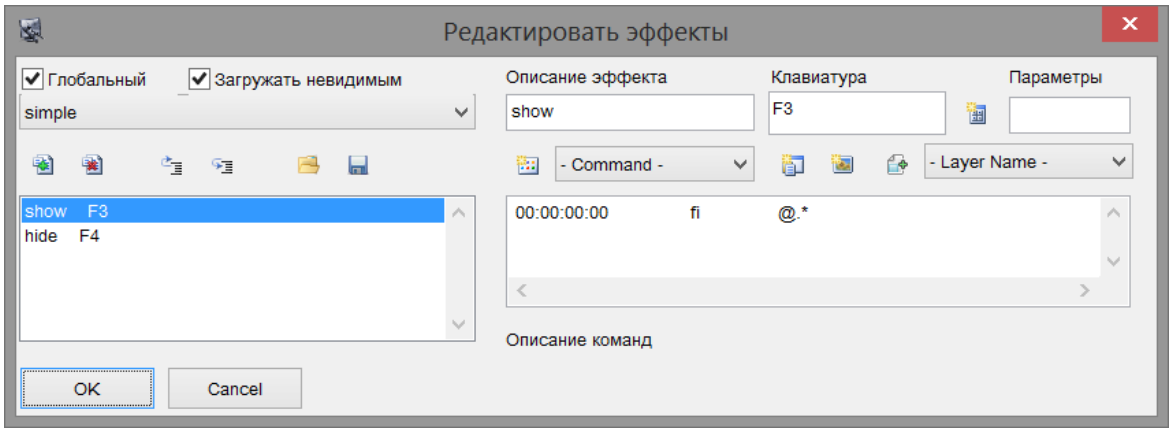
Содержимое эффектов по умолчанию записано в конфигурационном файле vged.cfx и его можно изменить.

Редактор эффектов

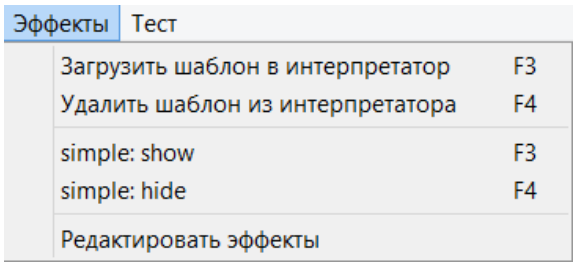
Чтобы использовать эффекты по умолчанию достаточно открыть редактор эффектов (меню "**Эффекты**" - "**Редактировать эффекты**") и поставить две галки "**Глобальный**" и "**Загружать невидимым**".

Теперь эти эффекты можно использовать через меню "**Эффекты**", в котором есть две строки "**show**" и "**hide**".

Эффектам можно назначать комбинации клавиш - клавиатурные шорткаты. Чтобы назначить эффекту шорткат, нужно сначала выбрать нужный эффект из списка, ткнуть мышом в поле "**Kbd Shortcut**", нажать нужную комбинацию клавиш на клавиатуре и нажать кнопку - "**Назначить клавишу**". Есть смысл назначать эффекту **show** кнопку **F3** и эффекту **hide** -- кнопку **F4**, чтобы они совпадали с командами "Загрузить шаблон..." и "Удалить шаблон...".



После назначения шорткатов эффектам они появляются и в меню эффектов:



Усложняем эффекты (simple3.tpl)

Вообще говоря, эффекты по умолчанию визуально ничем не отличаются от команд "**Загрузить шаблон...**" и "**Удалить шаблон...**". Пусть нам хочется показать шаблон плавным фейдом (микшером) за 10 кадров и убрать таким же фейдом. В эффекте **show** в команде fi можно после имени слоя указать количество кадров, за которое будет выполнен микшер:

```
00:00:00:00    fi    @.*    10
```

Также нужно изменить и эффект **hide**

```
00:00:00:00    fo    @.*    10
00:00:00:10    del   @
```

Обратите внимание, что начальное время выполнения команды **del** равно 10 кадрам, т.е. сначала выполняется команда **fo** -- убрать все слои за 10 кадров, и после этого (через 10 кадров) удаляется шаблон.

Если эффектам были назначены шорткаты F3 и F4, то по команде F3 все слои шаблона плавно проявятся за 10 кадров, а по команде F4 -- плавно растворятся за 10 кадров.

Индивидуальная работа со слоями (simple4.tpl)

В предыдущих примерах использовались команды микшера, которые применялись ко всем слоям шаблона одновременно. Если использовать команды применительно к конкретным слоям, то можно создавать более сложные эффекты.

Пусть нужно показывать слои шаблона simple в таком порядке -- сначала красную подложку bg[1], затем (через пять кадров) текст text[1], еще через пять кадров серую подложку bg[2] и еще через пять кадров -- текст text[2]. Тогда эффект **show** должен содержать такую последовательность команд:

```
00:00:00:00    fi    @.bg[1]      10
00:00:00:05    fi    @.text[1]    10
00:00:00:10    fi    @.bg[2]      10
00:00:00:15    fi    @.text[2]    10
```

Если убирать слои шаблона нужно в обратном порядке, то эффект **hide** может иметь такой вид:

```
00:00:00:00    fo    @.text[2]    10
00:00:00:05    fo    @.bg[2]      10
00:00:00:10    fo    @.text[1]    10
00:00:00:15    fo    @.bg[1]      10
00:00:01:00    del   @
```

Обратите внимание на время начала команды del -- оно должно быть равно времени выполнения последней команды эффекта (15) плюс длительность этого эффекта (10), т.е. в сумме 25 = 00:00:01:00.

Анимированные подложки с альфой c:\vgtutor\lower3rd

Немного о форматах и кодеках

Поскольку в этом примере используется анимированная подложка с альфа-каналом, то нужно пару слов сказать о тех форматах и кодеках, которые нужно использовать для представления анимации. В принципе **vgcast** понимает большое количество контейнеров (форматов файлов) и кодеков. Но на самом деле существует не так много кодеков для представления данных с альфа-каналом. Один из самых популярных кодеков -- это **Quicktime Animation** (или **QTRLE**), также можно использовать кодеки **PNG**, **TGA**, **TIFF** (если их поддерживает нужный контейнер). Однако все эти кодеки (не говоря уже об использовании некомпрессированных данных!) создают файлы достаточно большого размера и с высоким битрейтом. Если для стандарта SD это в общем то не такая большая проблема, то при переходе к высокому разрешению HD и размеры файлов и битрейт взлетают до небес, что накладывает жесткие требования на используемую дисковую подсистему.

Если же в сложных шаблонах используется одновременно несколько анимированных элементов, то требования к "железу" возрастают пропорционально.

Легче стало жить, когда появился кодек **Apple ProRes** с профилем 4:4:4:4:ALPHA (fourcc **ap4h**). Поэтому (не догма, но настоятельная рекомендация) желательно использовать файлы, в которых анимация записана с использование именно этого кодека. А для работы в HD это практически единственный удобный формат. Так файл, который используется в примере с кодеком ProRes в **пять** раз меньше, чем с кодеком PNG.

vgcast категорически не понимает анимацию, представленную как последовательность графических файлов. Т.е. если графика от дизайнеров поступила в таком формате, нужно конвертировать последовательность кадров в контейнер типа MOV или AVI. Для этого можно использовать программу **vgcvt.exe** из поставки **vgcast**.

О ширине и высоте кадра в анимации

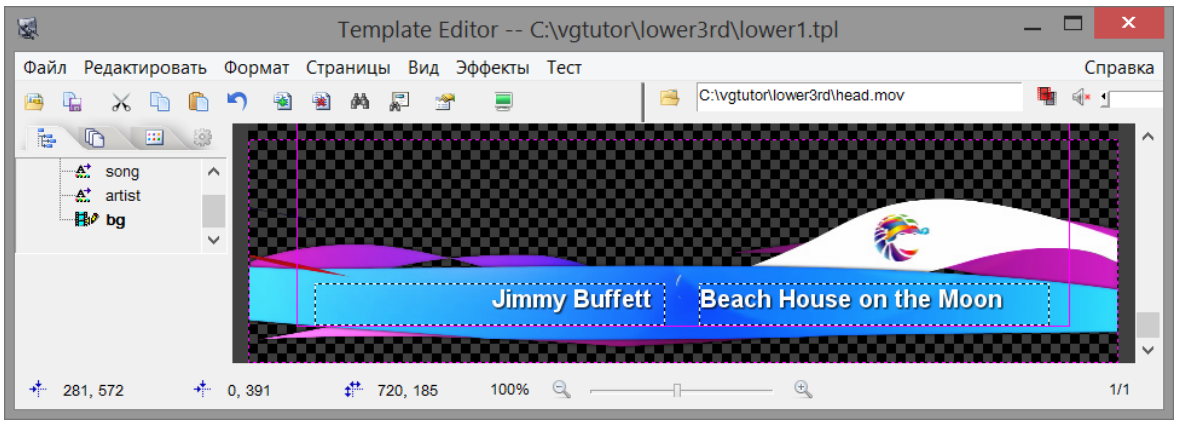
Любимое развлечение у дизайнеров -- это выдавать файлы совпадающие с размером кадра используемого стандарта, даже если реально используется только малая часть кадра. Опять таки, файл в примере имеет эффективный размер кадра 720x185 при полном размере кадра SD 720x576. Т.е. файл занимает всего одну треть от размера кадра (поэтому такие титры и имеют название lower third -- одна треть). Для декодера это не так уже и важно, но когда выполняется композитинг финального кадра, т.е. когда движок складывает все слои в один результирующий фрейм, операцию микширования нужно выполнять для каждого пикселя, вне зависимости от того, прозрачен этот пиксель или нет. Т.е. чем меньше площадь слоев, используемых для композитинга, тем менее загружен основной процессор, а значит можно успеть смикшировать большее число слоев.

Поэтому рекомендуется потратить немножко времени и обрезать все файлы с анимацией так, чтобы в них остались только области с непрозрачными пикселями. Это, опять таки, можно сделать с помощью утилиты **vgcvt.exe**, в ней есть режим поиска непрозрачных областей в автоматическом режиме и вычисления области кропа. Достаточно сохранить полученный файл с подходящим кодеком - и получаем анимацию без "бесполезных" областей.

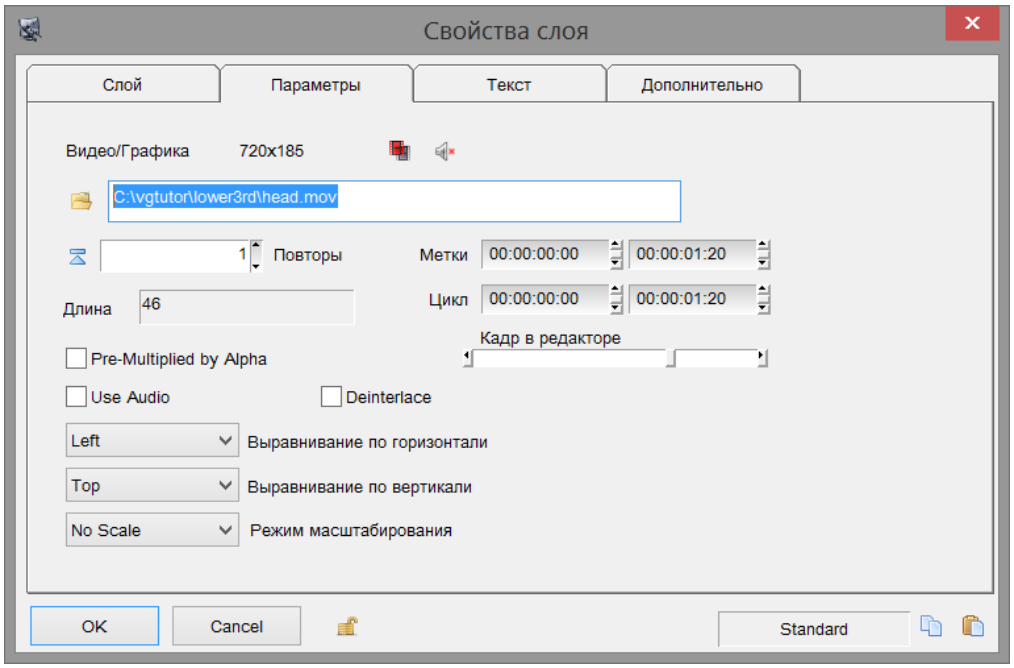
Типичная задача титрования (lower1.tpl)

Это наверное одна из самых распространенных задач титрования -- вывод титра, расположенного в нижней трети экрана. Титр выводится на анимированную подложку (анимация с альфа-каналом, каналом прозрачности), причем есть две последовательности -- одна используется для того, чтобы показать титр (**head.mov**), вторая - чтобы убрать его (**tail.mov**). На подложку нужно наложить текстовую информацию, например, имя артиста и название песни.

Создаем шаблон из трех слоев: **bg** для подложки, **song** - для названия песни, **artist** -- для имени артиста.



В свойствах слоя **bg** (его тип "**AnyMovie**") выбираем файл со стартовой анимацией



Совет: если потаскать мышкой ползунок "**Кадр в редакторе**", то в окне редактора будет показан выбранный кадр, т.е. можно выбрать именно тот кадр, по которому удобно располагать текстовые слои.

Слои **artist** и **song** -- текстовые слои, задаем нужные параметры текста (шрифт, цвет и пр.), и заполняем сам текст. Если попытаться использовать такой шаблон не создавая эффект show, то получится полная ерунда -- поскольку все слои станут непрозрачными (видимыми) одновременно, то надписи с именем артиста и названием песни появятся одновременно с началом анимированной подложки. А она начинается с прозрачности, и "раскрывается" только к концу первой секунды после начала эффекта. Причем сначала открывается поле для названия песни, а потом для автора.

Создаем эффект **show** (не забываем поставить галки "**Загружать невидимым...**" и "**Глобальный...**"):

```
00:00:00:00    fi    @.bg
00:00:00:23    fi    @.song  5
00:00:01:02    fi    @.artist 5
```

В первой команде делаем видимым слой с анимированной подложкой. Как только прозрачность слоя типа "**AnyMovie**" становится отличной от 0, начинается воспроизведение анимации из файла. Через 23 кадра коротким микшером (5 кадров) показываем слой **song**, и на 27 кадре таким же микшером показываем слой **artist**.

Когда файл со стартовой анимацией доиграется до конца, в эфире останется последний кадр подложки, и в таком виде титр можно держать в эфире сколько угодно, до выполнения команды **hide**. В принципе можно увести титр простым микшером, но у нас есть файл с закрывающей анимацией -- **tail.mov**, т.е. нужно каким-либо образом сыграть этот файл. Можно конечно создать для закрывающей анимации отдельный слой (например с именем **tail**,

по геометрии совпадающий со слоем **bg**), привязать к нему файл с **tail.tpl** и при выполнении эффекта **hide** спрятать слой **bg** и показать слой **tail**.

Но мы воспользуемся другим приемом -- применим команду ***замены*** содержимого слоя (***подстановки***). Это универсальная команда, она может применяться к слою практически любого типа, меняется только смыл параметра подстановки.

```
00:00:00:00    subst    @.bg    "C:\vgtutor\lower3rd\tail.mov"
00:00:00:12    fo       @.song  5
00:00:00:14    fo       @.artist 5
00:00:01:10    del      @
```

Первая команда эффекта и выполняет замену содержимого слоя **bg**. Первый параметр команды **subst** -- это полное имя слоя (имя шаблона плюс имя слоя). Второй параметр -- имя файла в двойных кавычках. Набирать имя файла в редакторе эффектов -- занятие скучное (да и ненадежное, можно запросто ошибок наделать), поэтому команду подстановки вводим как

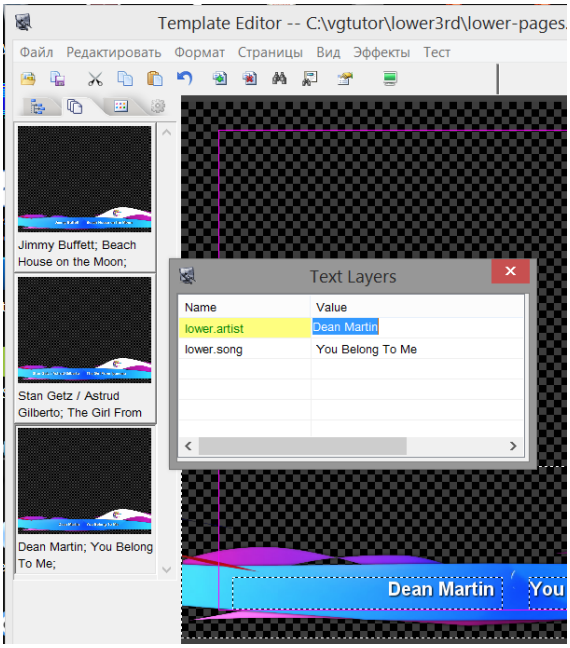
```
subst    @.bg    |
```

и установив курсор мыши на место параметра, ищем на инструментальной панели диалог эффектов кнопку "**Вставить имя файла**", открывается стандартный диалог выбора файла и выбранный файл вставляется на место курсора в команде.

При выполнении команды **subst** в эффекте **hide**, поскольку слой **bg** уже видимый воспроизведение кадров из файла **tail.mov** начнется сразу же после подстановки. Команды микшера **fo** в нужный момент времени уберут текстовые слои.

Одинаковый шаблон с разными текстовыми данными (lower-pages.vgd)

Для шаблонов типа "одна-треть" как правило нужно выдавать не один титр, а несколько, причем в титре меняются только текстовые данные. Чтобы хранить несколько страниц титров, нужно сделать документ редактора многостраничным (тип **vgd**). Для этого идем в меню "**Файл**" - "**Сохранить как**", в диалоге выбираем "**Тип файла**" как **VG Document (*.vgd)**, вводим новое имя файла, сохраняем.



В редакторе выбираем закладку "**Список страниц (CTRL+2)**". Щелкаем правой мышкой на пиктограмме нужной страницы (она, правда, пока ровно одна☺). Из выпадающего контекстного меню выбираем команду "**Копировать страницу**". Выбранная страница запоминается в буфере обмена. Опять вызываем контекстное меню, выбираем команду из группы "**Вставить страницу ...**", например "**Вставить после текущей...**". Редактор создаст новую страницу из шаблона, который был скопирован в буфер обмена.

Эту операцию (вставки страницы) можно повторить нужное количество раз. Осталось только поменять текст на каждой странице.

Если текст менять в свойствах слоя на закладке текст -- это достаточно неудобно, нужно выполнить много операций для изменения текста.

В редакторе есть специальный режим работы - редактирование только текстовых слоев. Этот режим включается и выключается из главного меню "**Вид**" - "**Текстовые поля (CTRL+T)**" или давим шорткат **CTRL+T**. На экране появляется табличка со списком текстовых слоев текущего шаблона и значениями в этих полях. Достаточно отредактировать нужный текст и он заносится в соответствующий шаблон. Окно для ввода текстовых полей можно разместить в любом месте экрана и выбрать для него необходимые размеры.

Опорные кадры и параметры слоев

Каждому слою соответствует набор параметров, которые определяют характеристики слоя -- прозрачность, координаты и пр... Управляя этими параметрами можно задавать динамическое поведение слоев, например, команда **fi** динамически меняет прозрачность слоя от 0 до 256 за указанное количество кадров. Есть в движке и ряд команд, которые меняют расположение объектов (**hslide**, **vslide**, **hmove**, **vmove**...) и расположение шаблонов (**tmove**, **torder**).

Но можно применять и другую, более универсальную, концепцию -- понятие опорных кадров.

Каждый слой в шаблоне имеет некоторое "**время жизни**", вообще говоря от момента загрузки шаблона в движок, до момента удаления шаблона. Поскольку в телевидении время -- это дискретная величина, то задавая **все** параметры слоя для каждого кадра можно **полностью** контролировать поведение слоя. Но задавать все параметры для каждого кадра -- занятие неблагодарное, да на самом деле и вовсе не нужное. Достаточно задавать параметры в некоторых **опорных** точках (**ключевых** кадрах, **keyframes**) и указать движку, по какому закону нужно вычислять все промежуточные значения этих параметров. Для реализации этой концепции в **vgcast** есть три команды работы с опорными кадрами

key
: изменение параметров объекта внутри слоя (**KEY**frame)

lkey
: изменение параметров всего слоя (**L**ayer**KEY**frame)

tkey
: изменение параметров всего шаблона (т.е. всех слоев в нем) (**T**emplate**KEY**frame)

Типы опорных кадров

Для каждого опорного кадра нужно задать время и параметры интерполяции. Время опорного кадра задает и тип кадра (**on-point** или **off-point**).

nNN, n+NN, n=EXPR, n+=EXPR
: ключевой кадр типа **on-point**, выполняется линейная интерполяция параметров. NN задает номер кадра, для которого указаны параметры. Если номер кадра задан в виде **n+**NN или **n+=**EXPR, то NN задает смещение относительно предыдущего ключевого файла.

fNN, f+NN, f=EXPR, f+=EXPR
: ключевой кадр типа **off-point**, выполняется сплайновая интерполяция параметров. NN задает номер кадра, для которого указаны параметры. Если номер кадра задан в виде **f+**NN или **f+=**EXPR, то NN задает смещение относительно предыдущего ключевого файла.

За определением опорного кадра следует один или несколько параметров слоя, разделенные пробелами или табуляторами.

Система координат

Для **vgcast** начало координат находится в левом верхнем углу экрана (или объекта для относительных координат), горизонтальная ось **X** направлена вправо, вертикальная ось **Y** направлена вниз. Координаты задаются в пикселях. Если мы говорим о координатах слоя или объекта -- то это координаты левого верхнего угла объекта.

Если используется команда **key**, то в ней указываются относительные координаты, т.е. координаты объекта, который привязан к слою относительно верхнего левого угла ***прямоугольника***, задающего слой. Этот же прямоугольник определяет и область вывода объекта, т.е. если объект или его часть выходит за рамки области вывода, то эти части объекта отсекаются.

Параметры слоев

tNN, t=NN, t=EXPR
: Задает прозрачность (**t**ransparency) слоя. Может изменяться в диапазоне от 0 (полностью прозрачный) до 256 (полностью непрозрачный).

  Может использоваться в командах **key** и **lkey**.

xNN, x=NN, x=EXPR
yNN, y=NN, y=EXPR

Задает координаты объекта. Координаты могут задаваться относительно начала экрана (команды **lkey** и **tkey**), либо относительно начала слоя (команда **key**). Произвольные положительные или отрицательные целые числа.

Может использоваться в командах **key**, **lkey** и **tkey**.

hNN, h=NN, h=EXPR
vNN, v=NN, v=EXPR

Задает коэффициент масштабирования объекта по горизонтали (**h**orizontal) или вертикали (**v**ertical). Поскольку все параметры слоя -- это целые числа, то задание коэффициента масштабирования имеет несколько непривычный диапазон значений, они задаются в единицах равных **0.1** процента, т.е. значение 1000 соответствует 100% (нормальные размеры объекта), 500 соответствует 50% (уменьшить вдвое), 2000 соответствует 200% (увеличить вдвое).

Может использоваться ***только*** в командах **key**.

wNN, w=NN, w=EXPR
sNN, s=NN, s=EXPR

Задает параметры шторки. **w** -- это положение шторки (**w**ipe), число от -256 до 256, а **s** -- "мягкость" (**s**oftness) края шторки, число от 0 до 256.

Может использоваться в командах **key** и **lkey**.

Интерполяция и опорные кадры c:\vgtutor\interpol

Каждому слою соответствует набор параметров, которые определяют характеристики слоя -- прозрачность, координаты и пр... Управляя этими параметрами можно задавать динамическое поведение слоев, например, команда **fi** динамически меняет прозрачность слоя от 0 до 256 за указанное количество кадров. Есть в движке и ряд команд, которые меняют расположение объектов (**hslide**, **vslide**, **hmove**, **vmove**...).

Но можно применять и другую, более универсальную, концепцию -- понятие опорных кадров.

Каждый слой в шаблоне имеет некоторое "**время жизни**", вообще говоря от момента загрузки шаблона в движок, до момента удаления шаблона. Поскольку в телевидении время -- это дискретная величина, то задавая **все** параметры слоя для каждого кадра можно **полностью** контролировать поведение слоя. Но задавать все параметры для каждого кадра -- занятие неблагодарное, да на самом деле и вовсе не нужное. Достаточно задавать параметры в некоторых **опорных** точках (**ключевых** кадрах, **keyframes**) и указать движку, по какому закону нужно вычислять все промежуточные значения этих параметров. Для реализации этой концепции в vgcast есть три команды работы с опорными кадрами

key -- изменение параметров объекта внутри слоя (**KEY**frame)

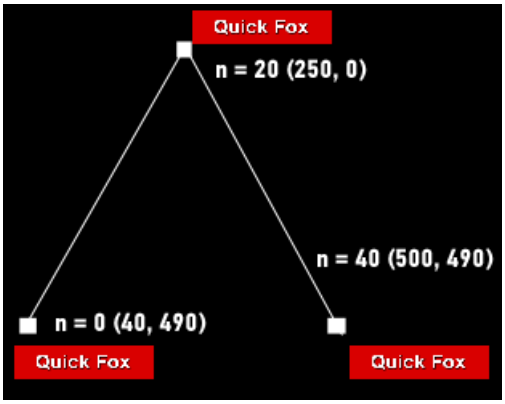
lkey -- изменение параметров всего слоя (L**KEY**frame)

tkey -- изменение параметров всего шаблона (т.е. всех слоев в нем) (T**KEY**frame)

В этом примере будет рассмотрена работа с ключевыми кадрами слоя.

Движение по прямой (key1.tpl)

Создаем простейший шаблон с двумя слоями -- подложка **bg** и текст **text**. Пусть стоит задача двигать эти слои по некоторой траектории, например начиная с начальной точки расположения до точки (**250**, **0**) и затем до точки (**500**, **490**), время движения от точки до точки -- 20 кадров.



Для такого движения достаточно задать три ключевых кадра: в момент времени 0 (начальная точка), 20 (промежуточная точка) и 40 (конечная точка). Поскольку движение происходит по **прямой**, т.е. линейно, то все три опорные точки имеют тип **n** (o**N** point). Для вычисления промежуточных значений между точками типа **n** выполняется **линейная интерполяция** всех промежуточных значений, и траектория слоя обязательно проходит через указанные опорные точки.

На самом деле первую точку можно не указывать, поскольку слой уже находится в ней.

С помощью команды **lkey** такая траектория описывается так:

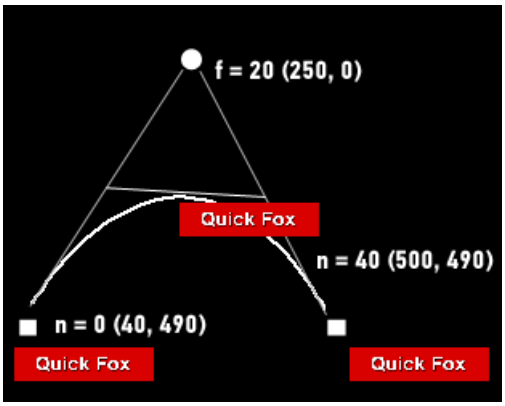
```
lkey    @.*    n=20 x=250 y=0    n=40 x=500 y=490
```

Желтым цветом в команде выделены опорные кадры и их параметры.

Движение по кривой (key2.tpl)

Если среднюю точку заменить на точку типа **f** (of**F** point), то между точками выполняется **нелинейная** интерполяция параметров, и траектория движения слоя становится **криволинейной**.

Крайние опорные точки (первая и последняя) всегда должны быть типа n, т.е. находиться на кривой. Изменяя расположение и количество промежуточных опорных точек можно существенно изменять траекторию движения слоя.

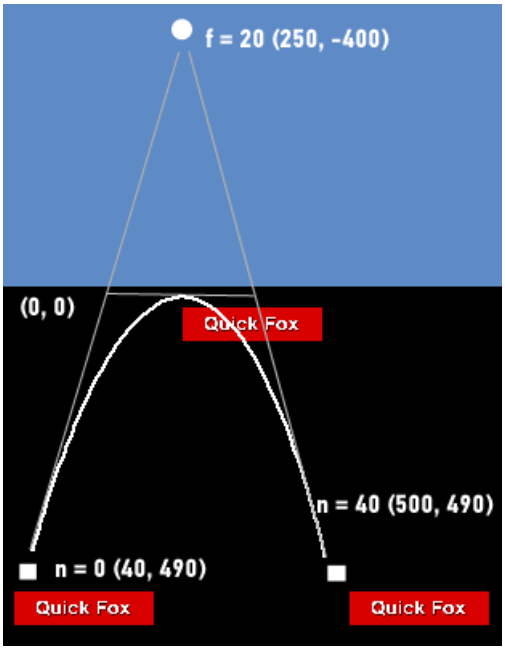


Точки типа **f** всегда лежать вне той кривой, которую они интерполируют. Для нелинейной интерполяции **vgcast** использует квадратичные полиномы Безье. Не вдаваясь в подробности можно сказать, что верхняя точка траектории доходит до середины отрезков, соединяющих опорные точки **n** и **f**.

Для задания такой траектории в команде **lkey** достаточно просто изменить тип промежуточной точки

lkey @.* f=20 x=250 y=0 n=40 x=500 y=490

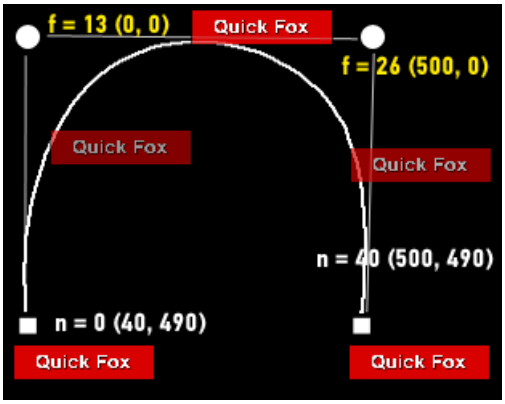
Изменяем траекторию (key3.tpl)



Чтобы сделать траекторию движения более крутой, так, чтобы прямоугольник слоя долетал до верхнего края экрана, нужно опорную точку **f** сдвинуть по координате **y** за пределы экрана, в отрицательную область.

lkey @.* f=20 x=250 y=-400 n=40 x=500 y=490

Кривая по нескольким промежуточным точкам (key4.tpl)



Если задать две промежуточные опорные точки типа f, то вид кривой также изменится:

lkey @.* f=13 x=0 y=0 f=26 x=500 y=0 n=40 x=500 y=490

Еще несколько примеров (key5.tpl - key8.tpl)

В этих примерах не используются новые команды, в них просто задается более длинная последовательность команд **lkey** для формирования "подпрыгивающего" слоя. В шаблонах key5.tpl и key7.tpl используется линейная интерполяция, в шаблонах key6.tpl и key8.tpl -- нелинейная.

Команды движения (move1.tpl)

Для перемещения объектов внутри слоя (изменения координат объекта) можно использовать специализированные команды движения по горизонтали и по вертикали -- **hslide**, **vslide**, **hmove** и **vmove**.

Команды слайдов имеют следующий формат

```
hslide  tpl.layer  speed_dur  from  to
vslide  tpl.layer  speed_dur  from  to
```

где

speed_dur

Задает либо скорость перемещения объекта либо длительность выполнения команды. Скорость задается как положительное целое число **NN** или как фраза **speed=NN**. Скорость указывает на сколько пикселей переместится объект за один полукадр. Если нужно задать продолжительность движения, то можно использовать фразу **dur=NN**, где **NN** - это количество кадров, за которое выполнится перемещение объекта.

from, to

Задают начальное и конечное положение объекта. Эти параметры могут быть числами **-1** (объект находится слева от области вывода), **0** (объект находится в области вывода) или **1** (объект находится справа от области вывода).

Команды движения имеют следующий формат

```
hmove  tpl.layer  speed_dur  from  to
vmove  tpl.layer  speed_dur  from  to
```

где

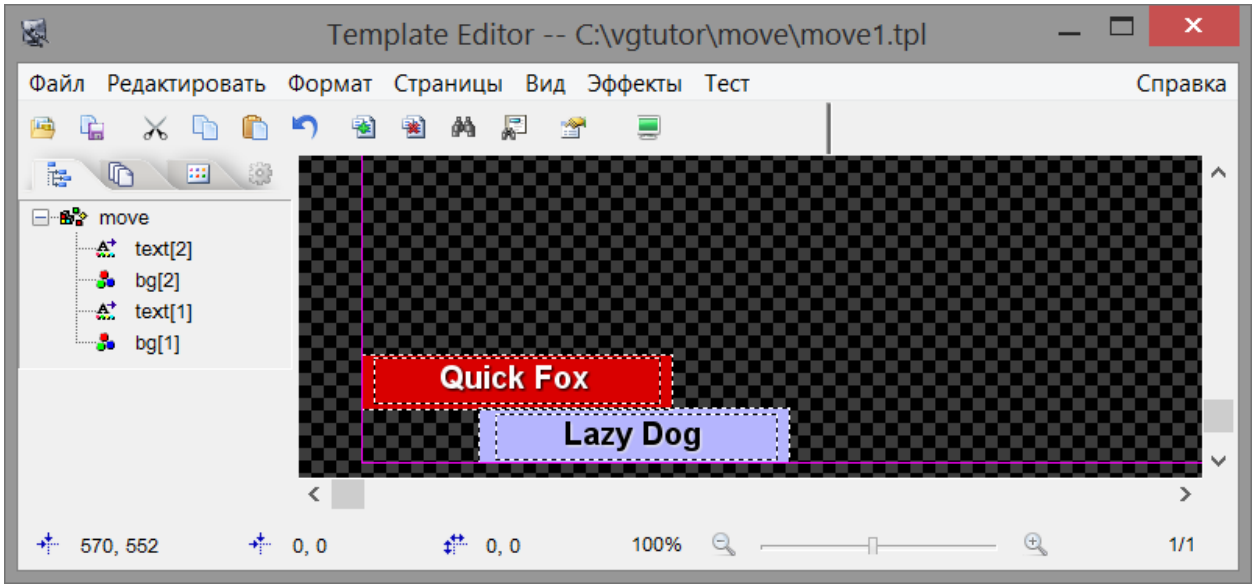
speed_dur

Задает либо скорость перемещения объекта либо длительность выполнения команды. Скорость задается как положительное целое число **NN** или как фраза **speed=NN**. Скорость указывает на сколько пикселей переместится объект за один полукадр. Если нужно задать продолжительность движения, то можно использовать фразу **dur=NN**, где **NN** - это количество кадров, за которое выполнится перемещение объекта.

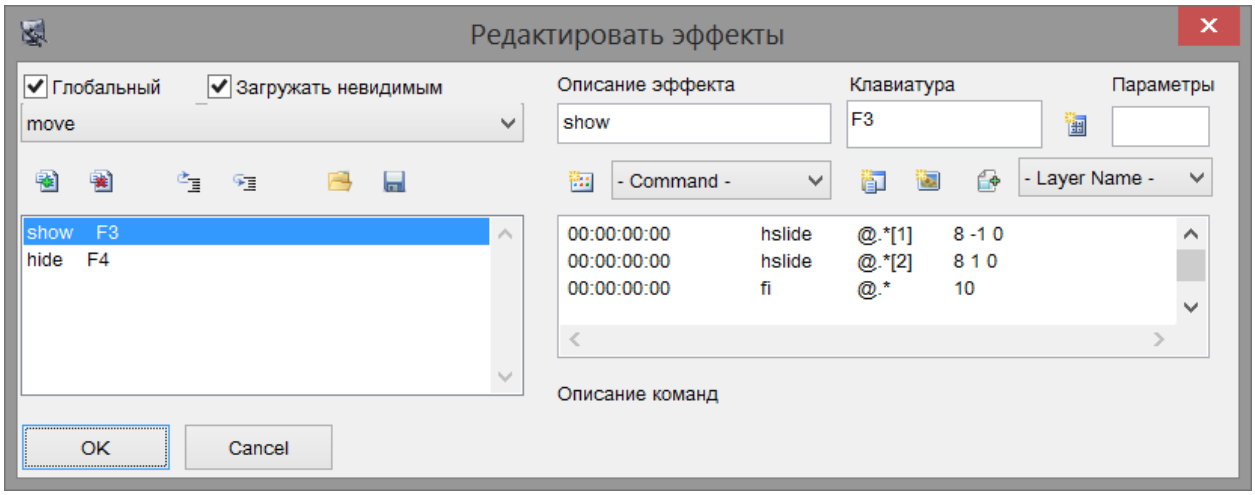
from, to

Задают начальное и конечное положение объекта. Эти параметры задают координаты объекта относительно области вывода, т.е. это произвольные положительные или отрицательные числа. Если параметр начинается со знака **=**, то значение определяется выражением ***=expr***.

Создаем шаблон, в нем четыре слоя, две подложки и два текстовых слоя.



Пусть нужно показать шаблон так, чтобы красная подложка с текстом (слои **bg[1]** и **text[1]**) выехала слева направо, а синяя подложка с текстом (слои **bg[2]** и **text[2]**) выехала справа налево. Для этого будем использовать команды горизонтальных слайдов



Рассмотрим команду для слоев **bg[1]** и **text[1]**

```
hslide   @.*[1]  8  -1  0
```

В качестве имени шаблона указываем символ **@** -- имя текущего шаблона, а в качестве имени слоя -- магическую последовательность знаков ***[1].** Символ ***** -- это символ заменитель, он совпадает с любой последовательностью знаков, поэтому в нашем случае такая фраза совпадет с любым именем слоя который заканчивается символами **[1]**, т.е. вся команда будет применена к слоям **bg[1]** и **text[1]**. Вместо такой команды можно было бы записать две команды с точными именами слоев:

```
hslide   @.bg[1]   8  -1  0
hslide   @.text[1] 8  -1  0
```

Параметр 8 задает скорость движения объекта, -1 означает что объект в начале движения находится слева от области вывода, а 0 означает что объект в конце движения станет в область вывода. Вторая команда применяется к слоям, имена которых заканчиваются символами **[2]**, т.е. к слоям **bg[2]** и **text[2]**, а движение задается справа налево (параметры 1 0). Последняя команда эффекта задает изменение прозрачности всех слоев (объектов).

Для эффекта **hide** зададим движение слоев в обратном порядке, и воспользуемся для этого командой **hmove**.

```
00:00:00:00   hmove  @.*[1]  dur=10  =0  =-wd(@.bg[1])
00:00:00:00   hmove  @.*[2]  dur=10  =0  =wd(@.bg[2])
00:00:00:10   del    @
```

Вместо скорости движения объекта задаем длительность **dur=10** (десять кадров), начальная координата равна 0 (знак равенства перед нулем можно опустить, это выражение которое вычисляет константу), а вот конечная задается выражением, в котором используется встроенная функция **wd(..)** движка **vgcast**. Эта функция вычисляет ширину слоя, который указан как параметр функции. В принципе, можно было бы посмотреть ширину слоя в редакторе и подставить конкретные числа (в примере ширина слоев **bg[1]** и **bg[2]** равна 200). Но если поставить конкретные числа, то после изменения размеров слоя в редакторе придется менять значения параметров в командах эффекта, что крайне неудобно. При использовании функций, как бы мы не меняли геометрию слоя (его размеры) эффект всегда будет выполняться правильно.

Движения по опорным кадрам (move2.tpl)

Вообще говоря команды слайдов и перемещения это некий рудимент, оставшийся от самых старых версий движка. Их можно заменить командой изменения параметров слоев по опорным кадрам -- командой **key**.

Для выполнения таких же действий можно написать такой эффект **show**

```
key   @.bg[1]     n0 t=0 x=-wd(@.bg[1])    n10 t=256 x=0
key   @.text[1]   n0 t=0 x=-wd(@.text[1])  n10 t=256 x=0
key   @.bg[2]     n0 t=0 x=wd(@.bg[2])     n10 t=256 x=0
key   @.text[2]   n0 t=0 x=wd(@.text[2])   n10 t=256 x=0
```

В каждой команде задано два опорных кадра, начальный кадр в момент времени **n=0**, и конечный, в момент времени **n=10**, т.е. эффект будет выполняться за 10 кадров. Интерполируемых параметра два -- прозрачность меняется от **t=0** до **t=256**, и горизонтальная координата меняется от **x=-wd(...)** до **x=0**. Горизонтальная координата **-wd(...)** указывает, что весь объект будет расположен слева от области вывода, координата **wd(...)** -- что объект расположен полностью справа от области вывода, ну а координата 0 указывает, что координаты объекта и области вывода совпадают, т.е. объект попадет точно в область вывода.

Эффект **hide** описывается аналогично, только начальные и конечные параметры прозрачности и координат стоят в обратном порядке.

Движения слоев (move3.tpl)

Если нужно двигать не сам объект внутри слоя (области вывода), а весь слой, то вместо команды **key** нужно использовать команду **lkey**. Если нужно чтобы оба слоя "прилетели" по вертикали из-за верхнего края экрана, то эффект **show** может иметь такой вид:

```
00:00:00:00    lkey    @.bg[1]      n0 t=0 y=-ht(@.bg[1])    n10 y=476 t=256
00:00:00:00    lkey    @.text[1]    n0 t=0 y=-ht(@.text[1])  n10 y=478 t=256
00:00:00:10    lkey    @.bg[2]      n0 t=0 y=-ht(@.bg[2])    n10 y=510 t=256
00:00:00:10    lkey    @.text[2]    n0 t=0 y=-ht(@.text[2])  n10 y=512 t=256
```

Для команды **lkey** координаты задают **абсолютные** значения, т.е. выражение **ht(@.bg[1])** вычислит значение 36, и координата y будет установлена в -36. Это означает, что весь слой будет расположен выше верхней границы экрана. Поскольку используются опорные точки типа **n** (on-points) то выполняется линейная интерполяция параметров, и слоя за 10 кадров прилетит из-за верхней части экрана в точку с координатой **y=476**.

В эффекте hide также будем использовать движение по опорным точкам, только по горизонтали, т.е. все слои со своей текущей позиции улетят за правую границу экрана.

```
00:00:00:00    lkey    @.*     n10 t=0 x=720
00:00:00:10    del     @
```

Поскольку начальная опорная точка **не** задана, то будут использованы текущие значения всех параметров слоев. В конечной опорной точке задана координата **x=720**, что для формата PAL задает координату пикселя за правой границей экрана. Имя слоя **@.*** указывает, что команда будет применена ко **всем** слоям текущего шаблона.

Движения слоев по кривой (move4.tpl)

В последнем примере вместо линейной интерполяции используется нелинейная, т.е. слои летают по криволинейной траектории. Использование нелинейной интерполяции описано в разделе Интерполяция, поэтому просто приводим команды для эффектов **show** и **hide**

```
lkey  @.bg[1]    n0 t=200 x=500 y=-ht(@.bg[1]) f5 t=200 x=600 y=500 f10 t=200 x=0 y=0 n15 x=40 y=476 t=256
lkey  @.text[1]  n0 t=200 x=500 y=-ht(@.text[1]) f5 t=200 x=600 y=500 f10 t=200 x=0 y=0 n15 x=40 y=478 t=256
lkey  @.bg[2]    n0 t=200 x=500 y=-ht(@.bg[2]) f5 t=200 x=600 y=500 f10 t=200 x=0 y=0 n15 x=126 y=510 t=256
lkey  @.text[2]  n0 t=200 x=500 y=-ht(@.text[2]) f5 t=200 x=600 y=500 f10 t=200 x=0 y=0 n15 x=126 y=512 t=256
```

Для удобства восприятия начальные опорные точки выделены желтым, конечные -- зеленым, а промежуточные (все они типа off-points) -- синим.

Эффект **hide** почему-то всегда получается проще☺

```
00:00:00:00    lkey    @.*     f5 x=360 y=0 n10 x=720 y=576
00:00:00:20    del     @
```

Команду применяем сразу ко всем слоям, и задаем только одну промежуточную опорную точку.

Масштабирование объектов c:\vgtutor\scale

Для изменения размеров объектов (масштабирования) используется команда **key** с параметрами **h=NN** и **v=NN**. Изменение масштаба позволяет получить довольно интересные эффекты, в том числе псевдотрехмерные переворачивания слоев, вращения и пр.

Масштабирование по вертикали (scale1.tpl)

Используем такой же шаблон, как и в примере с движением объектов, со слоями **bg[1]**, **text[1]** и **bg[2]**, **text[2]**. В эффекте **show** задаем такую последовательность команд:

```
00:00:00:00   key   @.bg[1]     n0 v=0 t=0   n10 v=1000 t=256
00:00:00:00   key   @.text[1]   n0 v=0 t=0   n10 v=1000 t=256
00:00:00:00   key   @.bg[2]     n0 v=0 t=0   n10 v=1000 t=256
00:00:00:00   key   @.text[2]   n0 v=0 t=0   n10 v=1000 t=256
```

Если внимательно присмотреться, то видно, что команды отличаются только именами слоев, и все это можно заменить одной командой ☺

```
00:00:00:00   key   @.*         n0 v=0 t=0   n10 v=1000 t=256
```

В этой команде меняется два параметра слоя -- прозрачность от 0 до 256 и вертикальный масштаб от 0 до 1000. Вспоминаем, что масштаб задается в десятых долях процента, т.е. вертикальный масштаб меняется от 0% до 100%. Остальные параметры слоев (в том числе и начальные координаты объектов) не изменяются, т.е. координаты X и Y остаются равными 0. В результате получается эффект появления слоев сверху вниз с вертикальным масштабированием.

Для эффекта **hide** применим другую последовательность команд.

```
00:00:00:00   key   @.text[2]   n0 y=0 t=256 v=1000   n10 t=0 v=0 y=ht(@.text[2])
00:00:00:00   key   @.bg[2]     n0 y=0 t=256 v=1000   n10 t=0 v=0 y=ht(@.bg[2])
00:00:00:00   key   @.text[1]   n0 y=0 t=256 v=1000   n10 t=0 v=0 y=ht(@.text[1])
00:00:00:00   key   @.bg[1]     n0 y=0 t=256 v=1000   n10 t=0 v=0 y=ht(@.bg[1])
00:00:00:10   del   @
```

Здесь меняем уже три параметра -- прозрачность меняется от 256 до 0, вертикальный масштаб от 100% до 0%, и вертикальная координата начала объекта меняется от 0 до значения высоты слоя. Поскольку меняется координата начала объекта, то слои сворачиваются в направлении сверху вниз.

Переворот по горизонтали (scale2.tpl)

Эффект **show** оставляем без изменений, но модифицируем эффект **hide** так, чтобы слои сворачивались по горизонтали, причем не в край слоя, а в его центр:

```
00:00:00:00   key   @.text[2]   n0 x=0 t=256 h=1000 n10 t=0 h=0 x=wd(@.text[2])/2
00:00:00:00   key   @.bg[2]     n0 x=0 t=256 h=1000 n10 t=0 h=0 x=wd(@.bg[2])/2
00:00:00:00   key   @.text[1]   n0 x=0 t=256 h=1000 n10 t=0 h=0 x=wd(@.text[1])/2
00:00:00:00   key   @.bg[1]     n0 x=0 t=256 h=1000 n10 t=0 h=0 x=wd(@.bg[1])/2
00:00:00:10   del   @
```

В отличие от предыдущего примера изменяется горизонтальный масштаб от 100% до 0% и координата X меняется от значения 0 до центральной точки слоя равной ширине слоя деленной пополам: **x=wd(...)/2**.

Увеличение из центра и уменьшение в центр (scale3.tpl)

Создаем совсем простой шаблон, состоящий из одного слоя типа **AnyMovie**. В параметрах слоя указываем какой файл связывается со слоем (**C:\vgtutor\graphics\apso.png**) и выбираем из комбо-бокса "**Режим масштабирования**" значение "**Keep Aspect**". Этот режим означает, что при изменении размеров слоя в редакторе (увеличении или уменьшении) объект (картинка в нашем случае) будет растягиваться или сжиматься с сохранением пропорций. Можно было бы и не заморачиваться с масштабом, но портреты собачек в примерах маленькие, а хочется показать эффект так, чтобы было понятно что собственно происходит.

Есть смысл обратить внимание на то, что мы связывали слой типа **Any Movie**, т.е. анимацию (видео) с обычным графическим файлом. Это совершенно легальное действие, графический файл интерпретируется как видеофайл, состоящий из одного кадра. Вообще говоря, слой типа **Still Image** можно вообще не использовать.

Формирует эффект показа так, чтобы объект появился из центра и увеличился до размеров слоя.

```
key     @.bg    n=0 x=wd(@.bg)/2 y=ht(@.bg)/2 v=0 h=0 n=25 x=0 y=0 v=1000 h=1000
```

В общем то ничего нового в этом эффекте нет, все отличие от предыдущих команд состоит только в том, что одновременно меняется четыре геометрических параметра объекта: начальные координаты и оба размера. Координаты начала объекта вычисляются как середина ширины и высоты объекта (**x=wd(...)/2** и **y=ht(...)/2**), конечные координаты равны 0 -- т.е. совпадают с началом слоя. Размер объекта меняется от 0% до 100%.

Эффект **hide** выполняет обратную задачу -- объект сворачивается в точку по центру слоя:

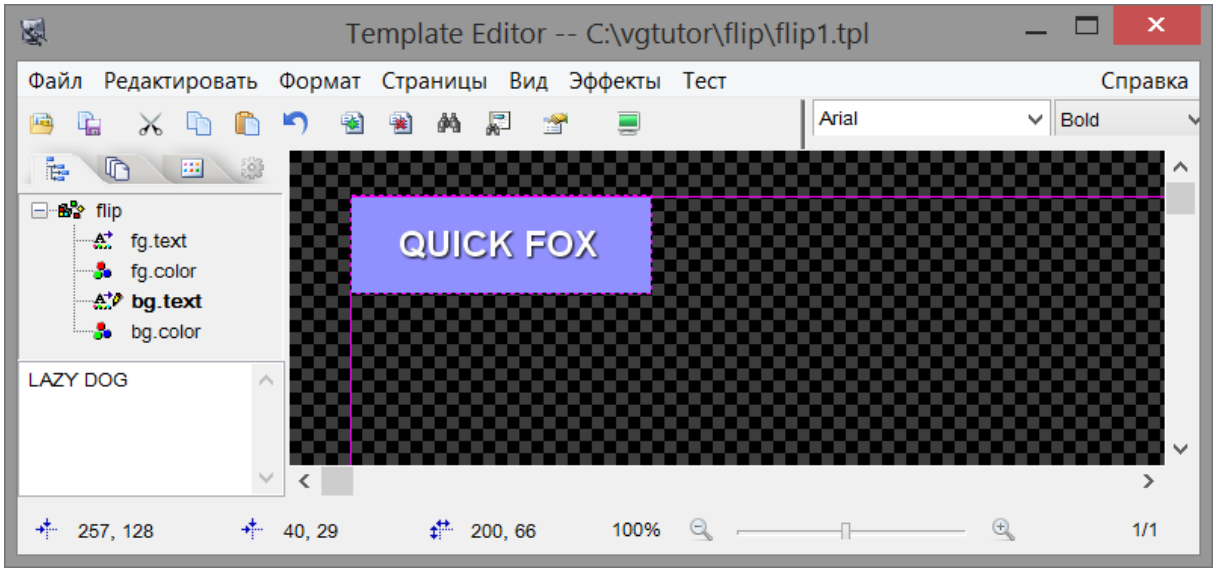
```
00:00:00:00   key    @.bg    n=0 x=0 y=0 v=1000 h=1000 n=25 x=wd(@.bg)/2 y=ht(@.bg)/2 v=0 h=0
00:00:01:00   del    @
```

Вращение объектов c:\vgtutor\flip

На самом деле никакого вращения объектов в трехмерном пространстве не происходит, но применение команд масштабирования позволяет получить эффект вращения объекта. В этом примере важно понимать именование объектов.

Вращение вокруг горизонтальной оси (flip1.tpl)

Создаем простой шаблон.



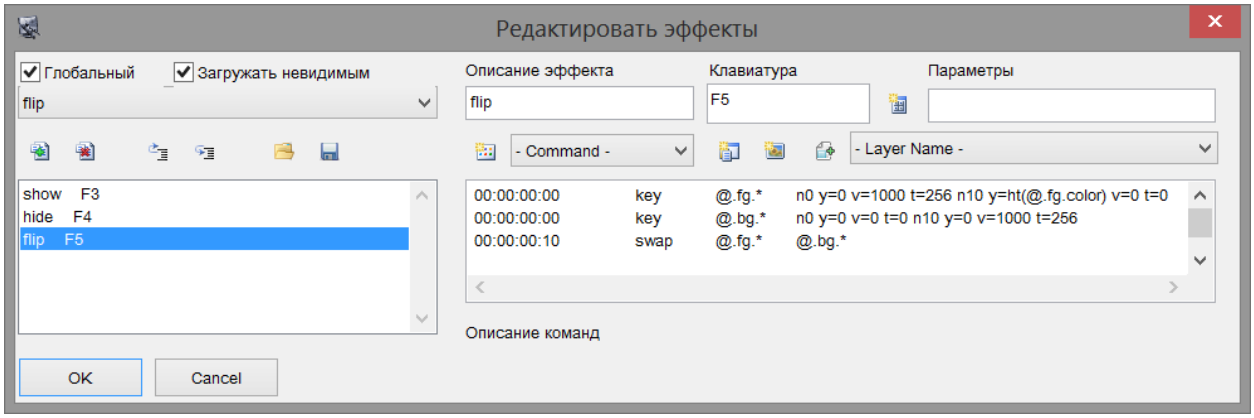
В шаблона создаем две группы слоев, слои с именами **bg.color** и **bg.text** -- это "задний план" (background) и слои с именами **fg.color** и **fg.text** -- это передний план (foreground). В слой **bg.text** вводим текст "LAZY DOG", а в слой **fg.text** вводим текст "QUICK FOX".

Слои имеют одинаковую геометрию (размеры и координаты) и расположены друг над другом. В эффекте **show** просто показываем все слои микшером за 10 кадров.

```
00:00:00:00     fi      @.*     10
```

Поскольку слои fg.* расположены над слоями bg.*, то все слои с именами bg.* остаются невидимыми, поскольку они перекрыты верхними слоями.

Создаем новый эффект, называем его **flip**, для удобства назначаем ему клавишу **F5**.



В первой команде эффекта задаем изменение масштаба слоев **fg.*** от 100% до 0%, причем одновременно сдвигаем координату Y от 0 до высоты слоя. Также одновременно меняем прозрачность от 256 до 0. Все слои **fg.*** начинают сворачиваться к нижней границе слоя.

Одновременно запускаем вторую команду, в которой слоям с именами **bg.*** меняем масштаб от 0% до 100%, прозрачность от 0 до 256, а координату Y задаем равной 0 как в начальной, так и в конечной точке. Все слои **bg.*** начинают разворачиваться от верхней границы слоя к нижней.

Поскольку эти две команды выполняются одновременно, т.е. один слой сворачивается, а второй -- разворачивается, выглядит все это так, как будто мы вращаем в трехмерном пространстве параллелепипед вокруг оси X, а на гранях параллелепипеда нанесены надписи "QUICK FOX" и "LAZY DOG".

Для просмотра этого эффекта в редакторе нужно сначала выполнить команду **show** (F3), а затем произвольное количество раз выполнять команду **flip** (F5). Ну и завершаем как всегда командой **hide** (F4).

Этот эффект **flip** написан в предположении, что перед началом эффекта слой **fg.*** открыт, а слой **bg.*** -- закрыт. Но после выполнения эффекта видимость слоев меняется, открытым становится слой **bg.***, а закрытым -- слой **fg.***. Т.е. если повторно выполнить эффект **flip**, он отработает неправильно! Казалось бы, нужно задавать второй эффект переворота, практически такой же как и эффект **flip**, только поменять в нем имена слоев. Но это скучно и неинтересно, поскольку вдобавок ко всему придется еще и вспоминать, какой эффект был выполнен перед текущим...

Переименование слоев

Чтобы разрешить этот конфликт в движке **vgcast** есть команда **swap** для обмена имен слоев. Синтаксис команды

```
swap    layer_1    layer_2
```

где *layer_1* и *layer_2* -- это полные имена слоев (т.е. имя шаблона за которым через точку следует имя слоя). Если в именах слоев используются символы заменители, то команда выполняется над каждым слоем в группе. Эта команда просто меняет имена слоев.

Состояния слоев меняются следующим образом, в начале выполнения эффекта (**черный** = видимый, **красный** = невидимый)

fg.color и **fg.text** - видимые **bg.color** и **bg.text** - **не**видимые

выполняем команды **key @.fg.* ....** и **key @.bg.* ....**

fg.color и **fg.text** - **не**видимые **bg.color** и **bg.text** - видимые

выполняем команду **swap @.fg.* @.bg.***. Имена слоев меняются местами:

bg.color и **bg.text** - **не**видимые **fg.color** и **fg.text** - видимые

В результате получили такое же состояние слоев, как и перед выполнение эффекта. А это значит, что можно опять безболезненно вызывать эффект **flip** и он отработает абсолютно правильно.

Если после выполнения эффекта **flip** заменить содержимое текста в слое **bg.text**,

```
subst @.bg.text ARTFUL WEASEL
```

то подстановка текста произойдет незаметно, поскольку слой **bg** невидимый, но при следующем выполнении эффекта появится уже новый текст.

Команду переименования удобно использовать во всех случаях, когда ведется работа с передним и задним планом, и все подготовительные действия выполняются на заднем плане, в скрытом режиме, а затем с помощью каких-либо эффектов выполняется смена планов, причем в процессе выполнения эффектов одновременно могут быть видны оба плана, и передний, и задний.

Добавляем цикл и картинку (flip2.tpl)

Этот пример ничем принципиально не отличается от предыдущего, просто добавили еще и картинку на грани **fg** и **bg** и будем использовать еще одну служебную команду **vgcast**-а -- команду формирования циклов **loop** в эффекте **show**.

```
00:00:00:00    fi      @.*     10
00:00:00:20    loop    -1  1000  efx  @.flip
```

Синтаксис команды

loop ***num_rep*** ***delay*** ***command***

num_rep

Количество повторений цикла. Если ***num_rep*** **> 0**, то цикл выполнится указанное количество раз. Если **< 0**, то цикл будет продолжаться бесконечно (пока его не остановят). Если ***num_rep*** **== 0**, то прекратится выполнение цикла

delay

Время выполнения цикла в миллисекундах. Вообще говоря это время должно быть большим (или равным) времени выполнения команды (или эффекта) указанного в последнем параметре цикла.

command

любая команда **vgcast**-а. Если нужно выполнить несколько команд, то придется оформить их как эффект и в цикле выполнять команду **efx**.

В примере в команде цикла указано, что цикл будет повторяться бесконечно (***num_rep*****=-1**), время выполнения цикл одна секунда (***delay*****=1000**) и в цикле будет вызываться эффект **flip**.

Поскольку время выполнения эффекта **flip** равно 10 кадрам или (в системе PAL) 10 * 40 = 400 мсек, то после выполнения эффекта переворота до начала следующего эффекта будет пауза длиной 1000 - 400 = 600 мсек. Если в параметрах команды цикла указать время, равное длительности эффекта (400 мсек)

```
00:00:00:20    loop    -1  400   efx  @.flip
```

то вращение плашки будет непрерывным, без пауз.

В эффекте **hide**

```
00:00:00:00    loop    0
00:00:00:00    fo      @.*     10
00:00:00:10    del     @
```

в первой команде прекращается выполнение цикла, дальше все знакомо.

При использовании команды цикла нужно учитывать тот факт, что в движке в каждый момент времени может выполняться только одна цикла. Если выполнить команду **loop** в то время, когда активен предыдущий цикл, то сначала прекратится выполнение предыдущего цикла, и только затем будет инициирован новый цикл.

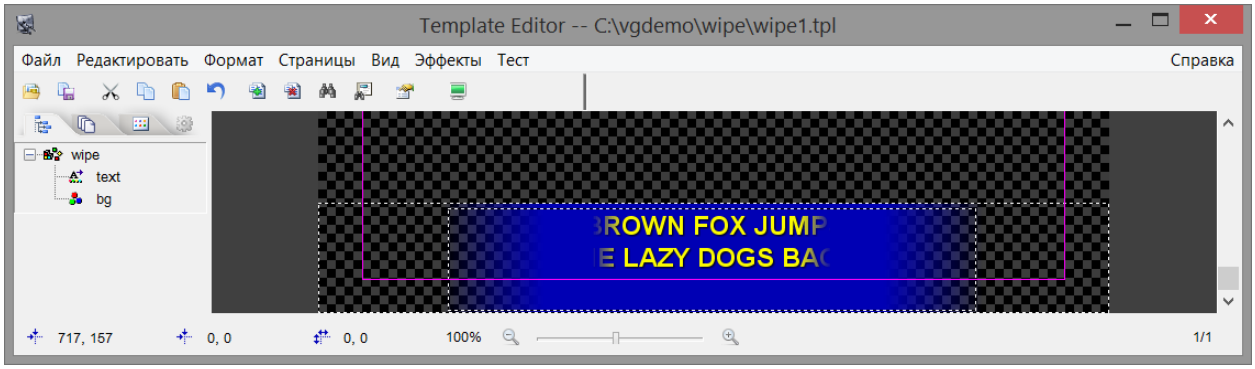
Работа со шторками c:\vgtutor\wipe

Шторки в **vgcast** -- это произвольные графические файлы, но для работы шторки используется только яркостная составляющая. Поэтому имеет смысл в качестве шторок использовать черно белые картинки. Шторка может быть назначена любому слою, и используя изменение параметров шторок по ключевым кадрам можно формировать достаточно необычные эффекты проявления и исчезновения слоев. Шторки, используемые в примерах должны находиться в папке **c:\vgdemo\gradients**.

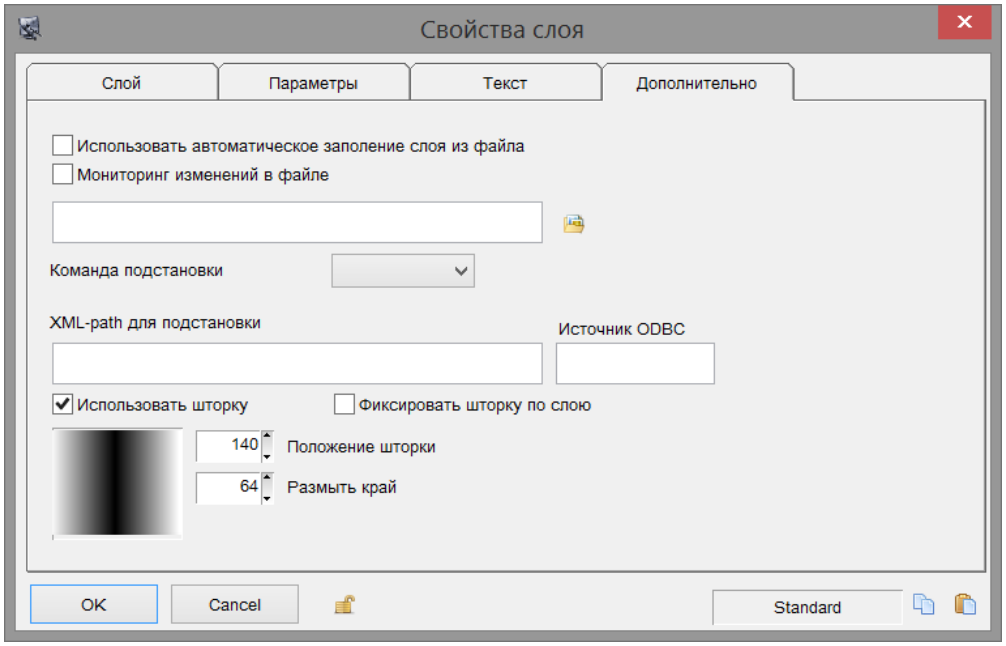
Использование шторок предполагает понимание работы по ключевым кадрам.

Простые шторки (wipe1.tpl)

Создаем шаблон с несколькими слоями, слой **bg** -- подложка типа сплошной цвет, слой **text** -- текстовый слой.



Чтобы связать со слоем шторку, нужно в параметрах слоя на закладке "**Дополнительно**" поставить галку на "**Использовать шторку**" и выбрать файл для шторки. Для выбора файла нужно ткнуть мышкой в прямоугольник под параметром "Использовать шторку", появляется окно выбора файлов с элементом предварительного просмотра.



Для примера выберем файл **BARNDR-H.png**. Эта шторка позволяет создать эффект "раскрывания" слоя от центра к краям. У шторки есть два параметра: "**Положение шторки**" (меняется от -256 до 256) и "**Размыть край**" (от 0 до 256). Меняя эти параметры можно сразу наблюдать, как изменяется слой с этой шторкой.

Значение параметра "**Положение шторки**" достаточно трудно описать, проще поэкспериментировать с ним в редакторе. Для выбранной шторки **BARNDR-H.png** изменение параметра от 0 до 256 приводит к тому, что слой "проявляется" от центра к краям, при изменении от 256 до 0 -- "растворяется" в обратном порядке, от краев к центру. Если этот параметр меняется от -256 до -1, то слой "проявляется" от краев к центру и т.д.

Назначаем эту шторку обеим слоям шаблона.

Если вывести такой шаблон без применения эффектов, то он будет выведен "как есть", т.е. ровно с теми положениями шторки, которые были заданы в свойствах слоя. Для получения динамики нужно использовать в эффектах команды, специфические для шторок.

В эффекте **show** откроем сначала слой **bg** то центра к краям, затем с небольшим смещением слой **text**, тоже от центра к краям. Не забываем поставить галки "**Глобальный**" и "**Загружать невидимым**".

```
00:00:00:00    wipe    @.* 0 64
00:00:00:00    fi      @.*
00:00:00:00    key     @.bg    n=25 w=256
00:00:00:10    key     @.text  n=25 w=256
```

В первой команде эффекта задаем начальное положение шторки. В этот момент все слои еще невидимы. Команда wipe имеет следующий формат

```
wipe    tpl.layer  pos  soft
```

Т.е. нужно обязательно указать имя шаблона и имя слоя, к которым применяется команда. Параметр ***pos*** задает положение шторки (число от -256 до 256). Значение **0** и **-256** указывает, что шторка полностью закрыта, т.е. слой связанный с этой шторкой невидимый. Второй параметр, ***soft***, задает мягкость края шторки, это число от 0 до 256.

Итак, в мы в эффекте для всех слоев указываем, что шторки закрыты (все слои невидимы) и мягкость шторки устанавливаем в 64.

Следующая команда **fi @.*** устанавливает значение прозрачности слоев в 256 (когда шаблон загружался с атрибутом "Загружать невидимым" прозрачность всех слоев была установлена в 0). Но поскольку шторки уже закрыты, то все слои по прежнему остаются невидимыми.

Следующая команда задает динамику изменения положения шторки

```
key     @.bg    n=25 w=256
```

В ней задан только один ключевой кадр **n=25** и в этом ключевом кадре задан параметр **w=256** -- значение положения шторки. Команда **key** выполнит интерполяцию всех промежуточных значений, начиная с текущего (оно равно 0, поскольку задано в команде **wipe**) до значения 256, и за 25 кадров продвинет шторку из положения 0 (полностью закрытая) в положение 256 (полностью открытая). Аналогичную команду задаем для слоя **text**, но указываем, что она начнется через 10 кадров после начала слоя **bg**.

Этот же эффект можно задать и другой последовательностью команд (эффект **showkey**):

```
00:00:00:00    key     @.bg    n=0 w=0 t=256     n=25 w=256 t=256
00:00:00:10    key     @.text  n=0 w=0 t=256     n=25 w=256 t=256
```

Для наглядности первый ключевой кадр выделен желтым цветом, а второй -- зеленым. Здесь используется тот факт, что в каждом ключевом кадре можно задавать несколько параметров слоя, в том числе и его прозрачность, т.е. в момент времени 0 (**n=0**) заданы параметры **t=256** -- установить начальную прозрачность слоя в 256 (сделать непрозрачным, видимым) и **w=0** -- начальное положение шторки. Во втором ключевом кадре **n=25** прозрачность не меняется **t=256**, а конечное положение шторки устанавливается в **w=256** -- полностью открытая шторка. Поскольку мы использовали ключевые кадры типа **n**, то выполняется линейная интерполяция параметров от начальных к конечным значениям

В эффекте **hide** задаем обратное направление для движения шторок:

```
00:00:00:00    key     @.text  n=0 w=-1 n=25 w=-256
00:00:00:10    key     @.bg    n=0 w=-1 n=25 w=-256
00:00:01:10    del     @
```

Т.е. слои будут растворяться от центра к краям.

Шторки и вывод рейтинга (wipe2.tpl)

Используя шторки можно делать достаточно необычные реализации показа рейтингов, когда нужно отразить, например, результаты голосования или опросов в динамике. Пусть есть два графических файла

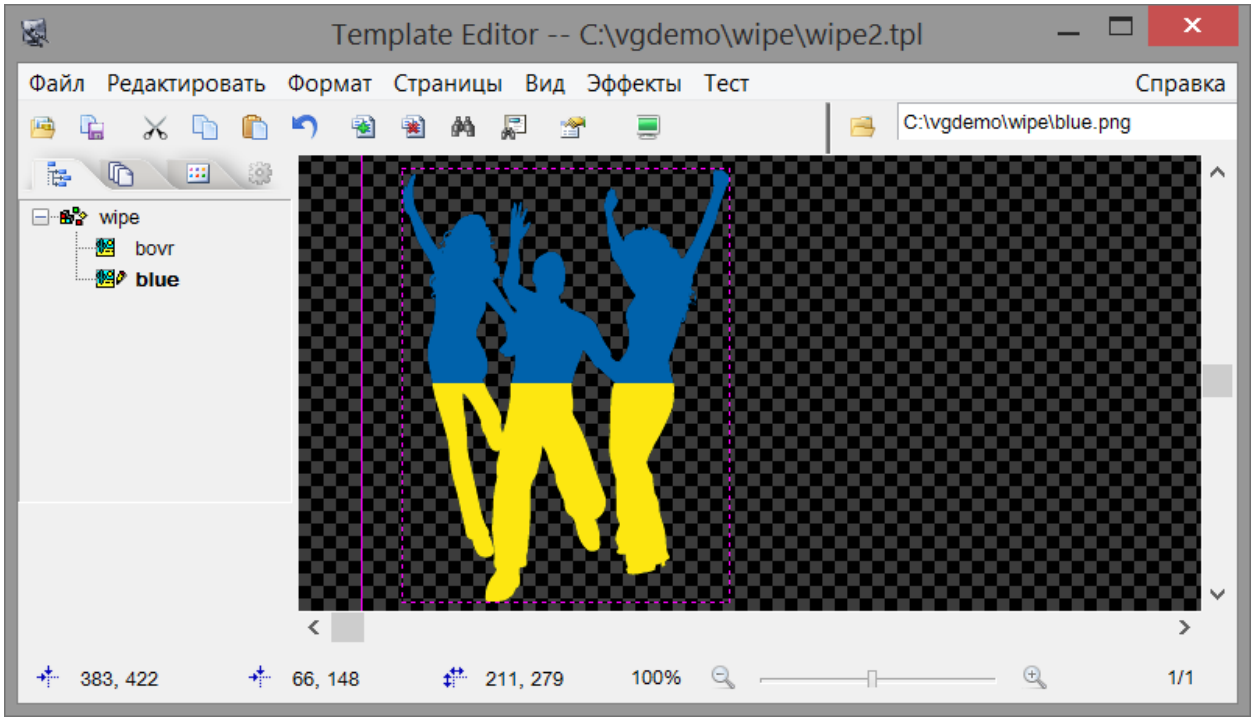
blue.png

blue-ovr.png

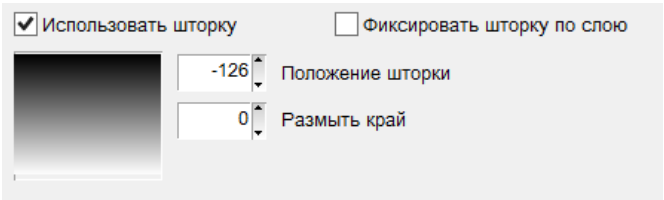
Это файлы с альфа-каналом, картинки на самом деле отличаются только цветом силуэтов.

Пусть нужно показать результаты голосования как заполнение синего силуэта желтым с низу вверх, пропорционально набранным процентам, т.е. результат голосования -- число от 0 до 100.

Создаем шаблон с двумя слоями типа "**Still Image**" (на саммом деле тип слоя не принципиален, это может быть и "**Any Movie**"), слой **blue** для синего силуэта и слой **bovr** для желтого. Слои располагаем друг над другом так, чтобы силуэты совпадали.



Для слоя **bovr** на закладке "**Дополнительно**" задаем вертикальную шторку



Для такой шторки положение -256 полностью закрывает слой **bovr** (желтый) а положение -1 полностью его открывает.

В эффекте **show** задаем динамику шторки

```
00:00:00:00     fi      @.blue  10
00:00:01:00     key     @.bovr  n=0 t=256 w=-256 n=25 t=256 w=-100
```

Первой командой показываем синий силуэт микшером за 10 кадров, затем через 1 секунду второй командой смещаем шторку из положения **w=-256** в положение **w=-100**.

В эффекте **hide** за 1 секунду закрываем шторку и удаляем шаблон.

```
00:00:00:00     key     @.bovr  n=25 w=-256
```

```
00:00:01:00     fo      @.*
00:00:01:00     del     @
```

Во всей этой истории есть один неприятный момент -- данные о результатах голосования мы получаем в процентах (числа от 0 до 100), а положение шторки меняется от -256 до -1, т.е. 0% соответствует -256, а 100% соответствует -1. Если нечеловечески напрячь мозг и вспомнить школьную арифметику, можно вывести такую формулу перевода процентов в положение шторки

w **=** ***pc******255/100 - 256**

где ***pc*** - проценты, а ***w*** -- положение шторки. Кто не верит -- можно подставить ну хотя бы крайние значения 0 и 100 и убедиться, что таки оно работает... Именно такую формулу будем использовать в следующем примере.

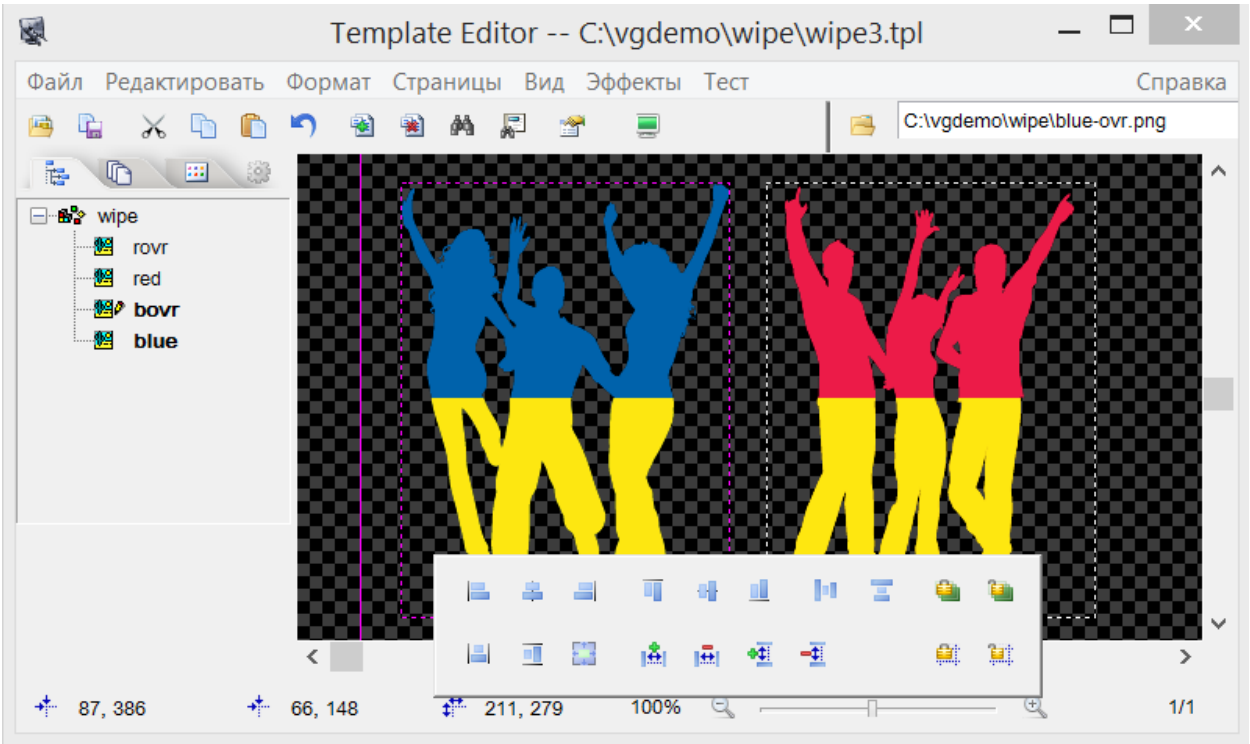
Шторки, вывод рейтинга и вычисления (wipe3.tpl)

Усложняем задачу. Пусть есть две команды, за которые ведется голосование, и нужно динамически отображать результаты голосования заполняя, соответственно, синий и красный силуэты желтым цветом. Результаты голосования получаем в процентах. Поскольку имеем две команды, есть очевидный факт, если за синюю команду проголосовало N%, то за красную 100-N%.

Добавляем в шаблон еще два слоя для красного силуэта -- **red** и **rovr**. Опять таки, эти слои должны совпадать по геометрии друг с другом.

Совет: для выравнивания слоев выбираем нужные слои в панели структуры шаблона -- зажимаем на клавиатуре **SHIFT** и тычем мышом сначала в один слой, потом во второй. Они выделяются жирным шрифтом (в примере **bovr** и **blue**). Не отпуская кнопку **SHIFT** давим правым мышом в выделенный слой в окне редактора. Выпадает контекстный диалог с пиктограммами выравнивания, давим "**Выравнивать левый...**" "**Выравнивать верх...**" "**Одинаковый размер...**"

То же самое можно делать через основное меню "**Формат**".

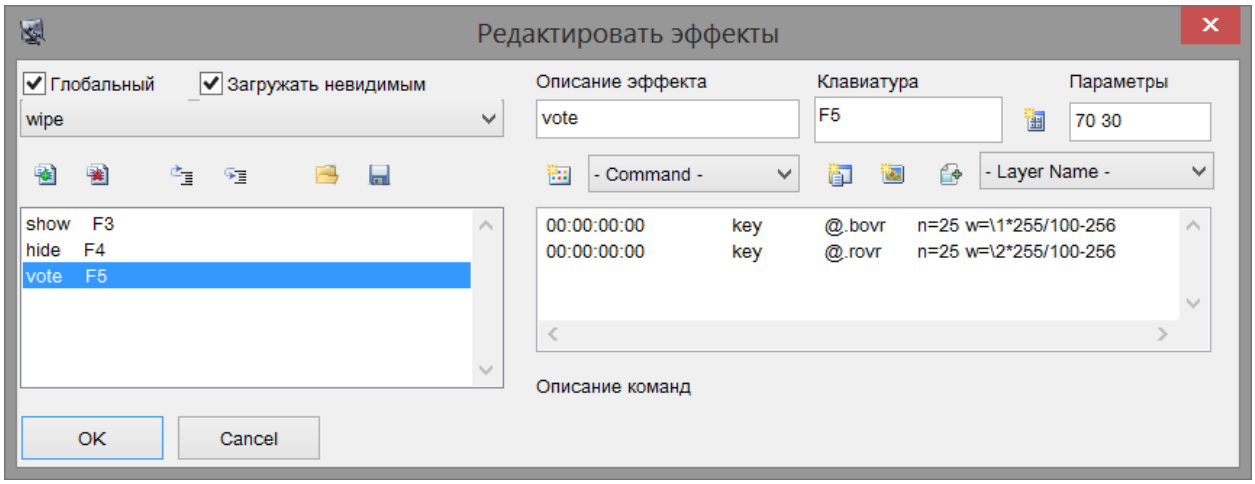


В эффекте **show** показываем только силуэты -- синий и красный. Ну и задаем начальное положение шторок.

```
00:00:00:00     wipe    @.bovr  -256 0
00:00:00:00     wipe    @.rovr  -256 0
00:00:00:00     fi      @.*     5
```

В эффекте **hide** просто убираем все.

Теперь добавим новый эффект, назовем его **vote**, в котором будет выполняться отображение результатов голосования.



В отличие от предыдущих эффектов, этот эффект -- параметрический. В качестве параметров выступают величины рейтинга в процентах, первый параметр для синего силуэта, второй параметр -- для красного. Значения параметров задаем в поле "**Параметры**", в нашем примере 70 (за синих) и 30 (за красных). Теперь пора вспомнить формулу пересчета процентов в положение шторки

w **=** ***pc******255/100 - 256**

В ключевых кадрах можно вместо численных значений атрибутов указывать выражения, т.е. вместо

key @.bovr n=25 w=-100

можно написать

key @.bovr n=25 w=\1*255/100-256
key @.rovr n=25 w=\2*255/100-256

В формуле значения процентов мы обозначали переменной ***pc***, а в эффекте указываем комбинацию знаков **\1** -- это означает, что в это место мы хотим подставить значение первого параметра. Соответственно, **\2** -- второй параметр и т.д.

Все выражение должно быть записано без пробелов, т.е. выражением считается вся последовательность знаков от знака **=** до первого пробела или конца строки.

Во всех случаях, когда мы в редакторе применяем какой либо эффект, интерпретацией эффекта занимается движок **vgcast**. Например, при выполнении эффекта **show** (как бы мы его не вызвали, через шорткат или через меню) движку отправляется команда

efx wipe.show

Поскольку в эффекте **wipe** были указаны параметры, то движку будет отправлена команда

efx wipe.vote 70 30

В самом эффекте движок выполнит подстановку параметров

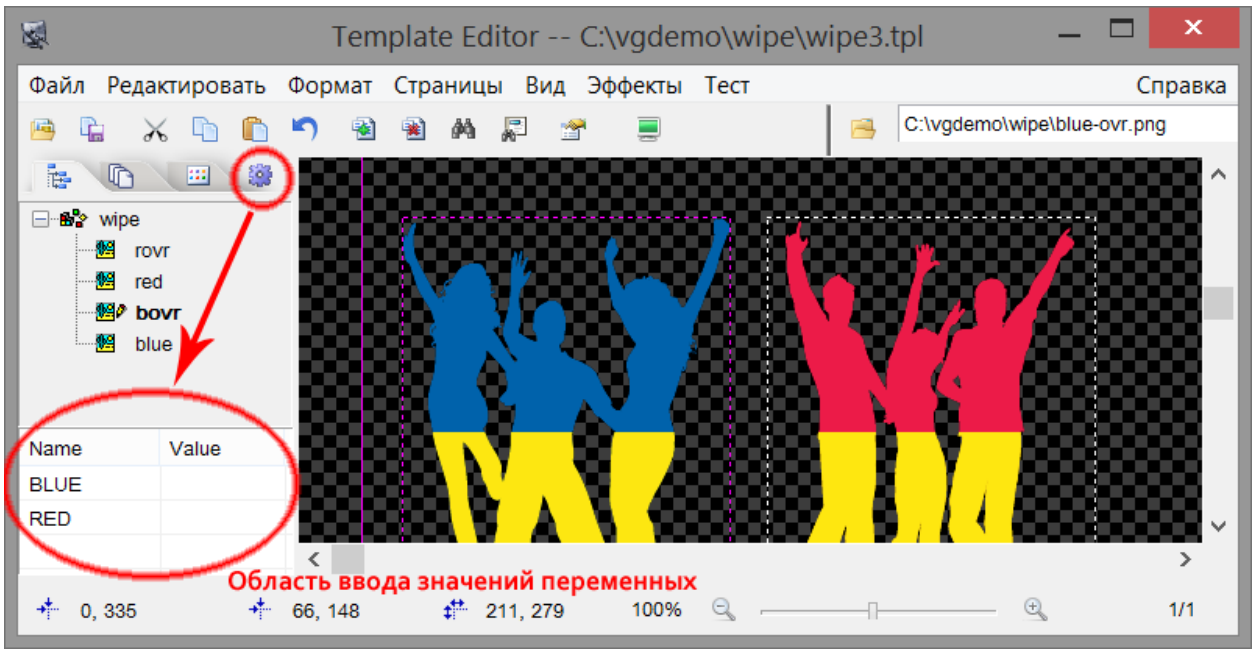
key @.bovr n=25 w=70*255/100-256
key @.rovr n=25 w=30*255/100-256

и вычислит выражения. Т.е. выполнение такого параметрического эффекта эквивалентно выполнению команд

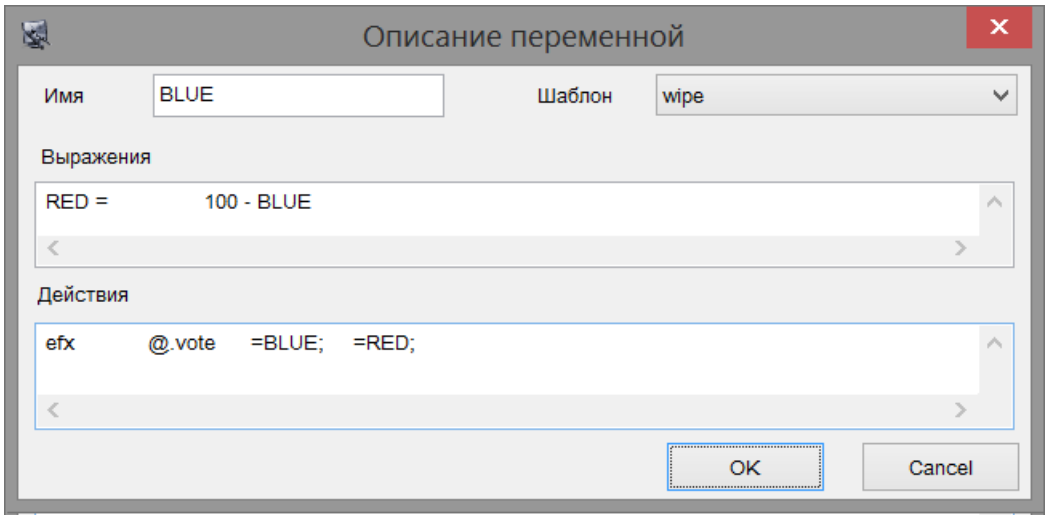
key @.bovr n=25 w=-78
key @.rovr n=25 w=-180

В общем то это уже неплохо, но сильно страдает оперативность ввода данных, чтобы задать значения процентов нужно вызвать редактор эффектов, выбрать эффект **vote** и поменять его параметры, и только потом вызывать эффект **show** и **vote**....

На самом деле в vgcast есть механизм, который позволяет оперативно вводить такие данные и сразу после ввода данных запускать на выполнение указанный эффект. Это механизм переменных. Для того, чтобы инициировать режим работы с переменными на закладках режимов редактора нужно выбрать режим **"Список переменных (CTRL+4)"**



Для описания и изменения переменных нужно выбрать нужную операцию в контекстном меню -- правая мышка в области переменных. Мы будем использовать две переменные **BLUE** для ввоза значений синей команды и **RED** для ввода значений красной команды. Рассмотрим описание переменной **BLUE** для ввода значений для синего силуэта.



Для переменной необходимо задать **имя**. Далее, с переменной может быть связано одно или несколько дополнительных выражений. Эти выражения удобно использовать когда применяются **взаимозависимые** переменные, когда значение одной переменной зависит от другой. В нашем случае они таки связаны, поскольку сумма значений голосов в процентах всегда составляет 100%. Итак в **поле выражений** вводим простую формулу **RED = 100 - BLUE**. Очевидно, что для переменной **RED** формула будет другой **BLUE = 100 - RED**.

В поле выражений каждая формула записывается на отдельной строке

Также с переменной в **поле действий** можно связать последовательность команд, которые будут выполняться при изменении значений переменных. В нашем случае это вызов эффекта vote с нужными параметрами, а именно со значениями переменных **BLUE** и **RED**.

efx @.vote =BLUE; =RED;

Обратить внимание, что при передаче параметров эффекту мы опять таки указываем не просто имена переменных, а выражения. Это означает, что в параметры эффекта подставятся уже **значения** переменных BLUE и RED, а не просто строки "BLUE" и "RED".

Когда описаны все переменные, связанные выражения и действия можно уже просто работать с получаемыми данными и оперативно менять картинку в эфире. Для ввода значений нужно просто ткнуть мышкой в поле **Value** в

таблице переменных, например в строке переменной BLUE. Поле таблицы переводится в режим ввода данных и достаточно ввести с клавиатуры нужное число, например 57. Завершить ввод числа можно клавишей **ENTER** или **TAB**, и тогда введенное значение присваивается переменной, выполняются вычисления всех связанных переменных (в нашем случае переменной RED присваивается значение 43), вычисленные значения отображаются в таблице переменных и запускается на выполнение соответствующий эффект -- в нашем примере эффект vote с параметрами 57 43.

Если ввод значения завершить клавишей **ESC**, то ввод данных игнорируется, связанные переменные не вычисляются и эффекты не запускаются.

Типичная последовательность действий для работы с переменными: выполнить эффект **show**, затем можно один или несколько раз вводить данные в нужные переменные, причем вводить новые значения можно подряд, и заполнения силуэтов будет выполняться от текущих значений к новым. И завершаем всю последовательность эффектом **hide**.

Ниже приведены скриншоты для разных значений переменных.

33 67 

88 12 

57 43 

Шторки и динамический текст (wipe4.tpl)

Напоследок еще один прием, который не имеет прямого отношения к шторкам, но часто используется во время показа результатов голосования. Пусть кроме заливки желтым цветом нужно еще показывать и сами значения количества голосов в процентах. Добавляем к шаблону два текстовых слоя **rpc** -- проценты для красного силуэта и **bpc** -- проценты для синего силуэта.

В принципе, можно было бы просто добавить в эффект **vote** две команды для подстановки значений процентов:

```
key     @.bovr  n=25 w=\1*255/100-256
subst   @.bpc   \1%
key     @.rovr  n=25 w=\2*255/100-256
subst   @.rpc   \2%
```

Но как-то некрасивенько получается, заливка меняется плавно, а значения чисел -- заменили и все. Хотелось бы получить **изменяющиеся** значения чисел, от текущего значения до нового. Для этого в движке есть команды **text**.

```
text    tpl.layer dur  value  format
```

где ***dur*** -- длительность эффекта в кадрах, ***value*** -- новое значение текстового слоя, к которому будет интерполировано текущее значение, и ***format*** -- необязательный параметр, который указывает, как будут отформатированы интерполированные значения.

В эффект **show** добавляем две команды, которые инициализируют работу с текстовыми слоями.

```
00:00:00:00     text    @.bpc   1 key 0   %.0d
00:00:00:00     text    @.rpc   1 key 0   %.0d
```

Т.е. обеим текстовым слоям присваиваем значение 0, а формат %.0d указывает, что если выводимое значение равно 0, то не показывать его вовсе. Поскольку длительность команды 1 кадр, никакой интерполяции не выполняется и присваивание происходит сразу. Т.е. когда выполнится команды **fi @.*** в текстовых слоях ничего показано не будет.

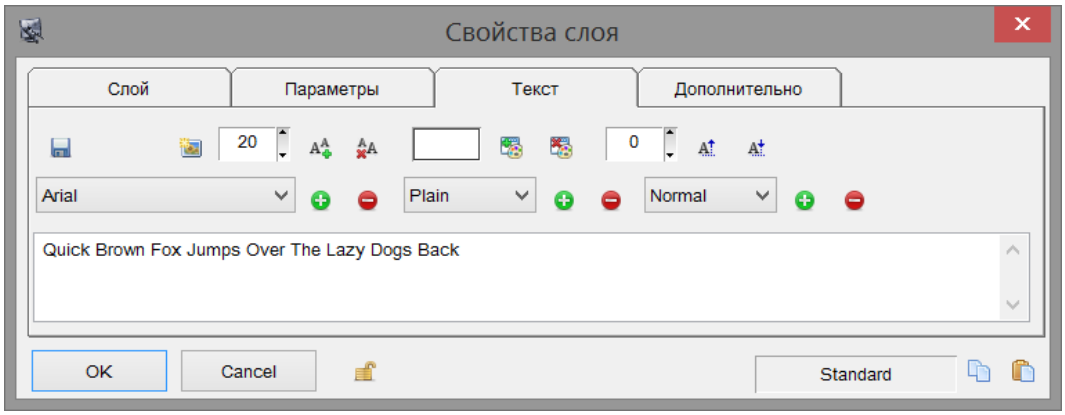
В эффект **vote** также добавляем две команды

```
00:00:00:00     text    @.bpc   25 key  \1   %.0f%%
00:00:00:00     text    @.rpc   25 key  \2   %.0f%%
```

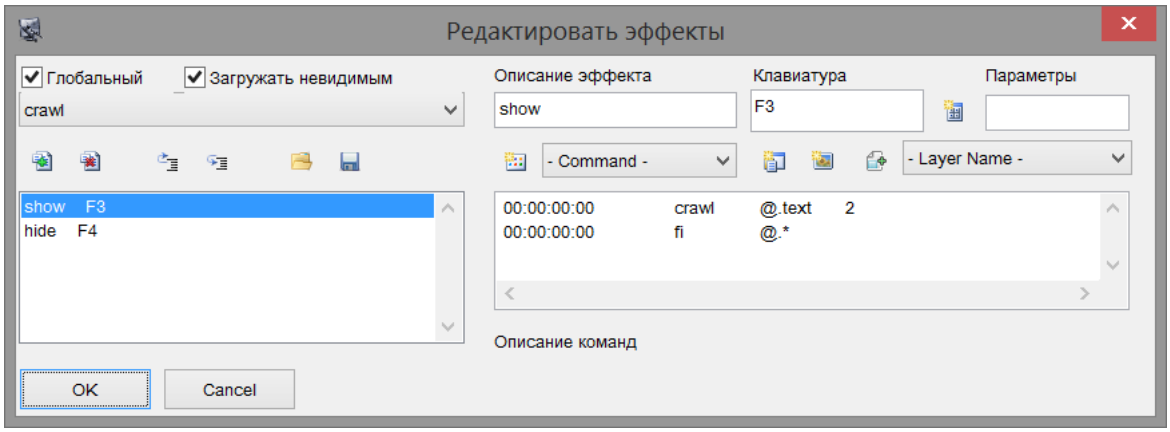
Т.е. за 25 кадров будут показаны значения первого параметра, интерполированные от текущего значения к новому. Указанный формат **%.0f** означает, что число будет сформатировано как целое, без дробной части. Двойной знак **%%** означает что к сформатированному числу будет приписан один знак **%**.

Простая бегущая строка (crawl1.tpl)

Создаем шаблон с двумя слоями, слой **bg** -- подложка, слой **text** -- собственно слой с текстом бегущей строки. В свойствах слоя **text** на закладке "**Текст**" вводим содержимое текстового слоя



Эффект бегущей строки задается в эффектах редактора: меню "**Эффекты**" - "**Редактировать эффекты**".



Ставим галки "**Глобальный**" и "**Загружать невидимым**". Для удобства назначаем эффекту **show** клавишу **F3**, эффекту **hide** -- клавишу **F4**. Для этого нужно ткнуть мышом в поле "**Клавиатура**" и ввести нужную комбинацию клавиш, затем нажать кнопку - "**Назначить клавишу**".

Команда **crawl** задает эффект бегущей строки. Формат команды

crawl **template.layer** ***speed num_rep***

где:

template -- имя шаблона
layer -- имя слоя

speed -- скорость движения строки. Может задаваться как:

- Целое число.
  Если число положительное -- строка движется справа-налево. Если число отрицательное -- строка движется слева-направо. Число указывает, на сколько пикселей продвинется строка за один полукадр.
- **speed=**expr
  В отличие от первого способа задания скорости можно задавать выражение.
- **dur=**expr
  Задает время, за которое строка пробежит по своей области вывода. Например, если задано **dur=**55:13, то строка пройдет через область вывода за 55 сек и 13 кадров.

num_rep -- количество повторений. Если не задано -- строка пробежит один раз. Если задано положительное целое число, строка пробежит указанное количество раз. Если задано значение **-1**, то строка будет прокручиваться бесконечное количество раз.

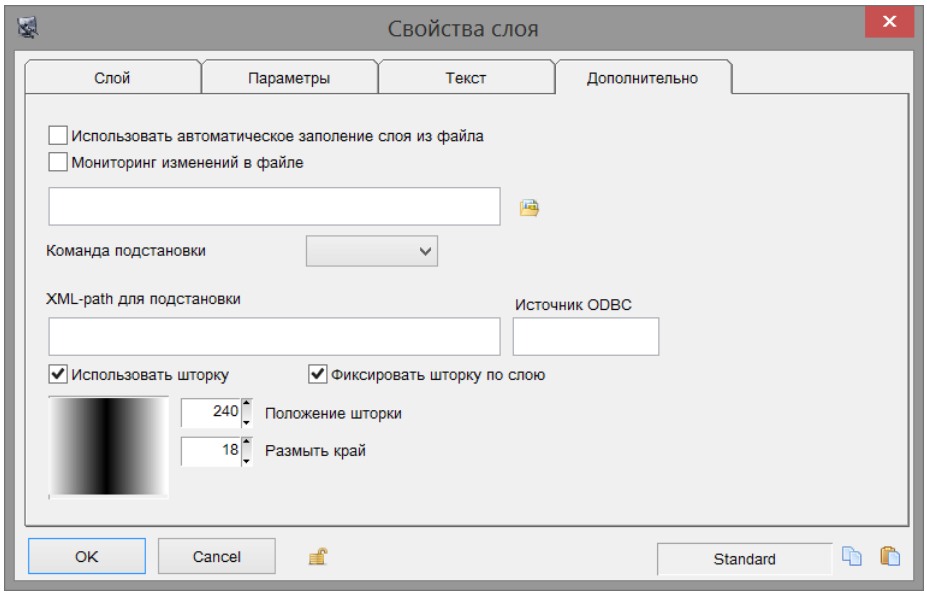
Закрываем редактор эффектов (OK). Давим F3 на клавиатуре. На синей подложке начинает двигаться бегущая строка.

Во всех командах если в качестве имени шаблона указан знак **@**, то в качестве имени шаблона используется текущий шаблон.

Поскольку в настройках эффекта задано "**Загружать невидимым**" нужно еще команда, которая покажет нужные слои шаблона (в нашем случае -- все). Это команда **fi** (fade in) во второй строке эффекта.

Бегущая строка с "мягким" краем (crawl2.tpl)

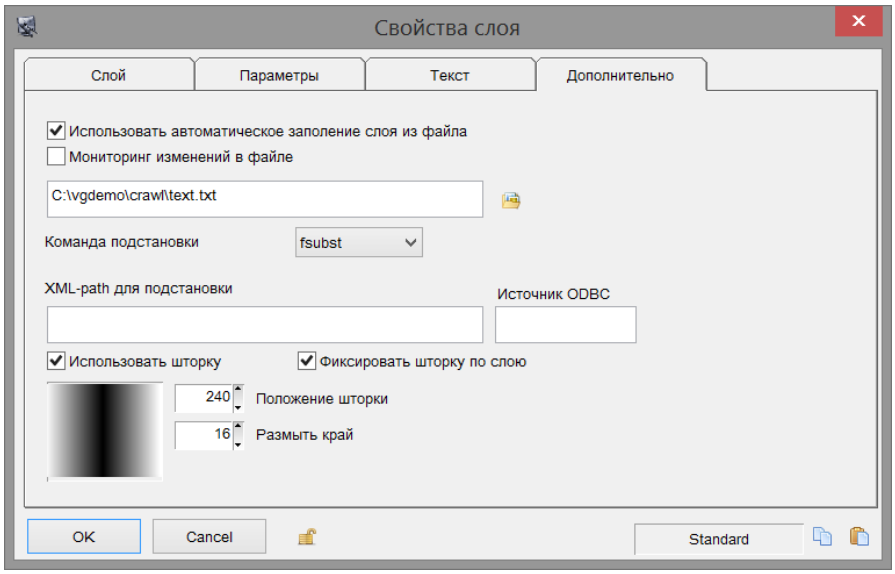
Если бегущая строка должна иметь "мягкие" края, нужно задать шторку, которая указывает как будут проявляться края бегущей строки. В свойствах слоя **text** идем на закладку "**Дополнительно**"



Ставим галки "**Использовать шторку**" и "**Фиксировать шторку по слою**". Для выбора файла шторки нужно ткнуть мышом в пиктограмму шторки. Открывается стандартный диалог выбора файлов. Шторка (Alpha Wipe) -- это произвольный графический файл. Задаем параметры "**Положение шторки**" (меняется от -256 до +256) и "**Размыть край**" (от 0 до 256).

Текст строки из файла (crawl3.tpl)

Если нужно, чтобы текст бегущей строки задавался не в самом шаблоне, а в некотором внешнем файле, то в свойствах слоя **text** идем на закладку "**Дополнительно**"



Ставим галку "**Использовать автоматическое заполнение слоя из файла**". Давим на кнопку и выбираем нужный текстовый файл (например, text.txt). В комбо-боксе "**Команда подстановки**" нужно выбрать **fsubst**.

Если текстовый файл имеет префикс кодировки **BOM**, то при чтении файла он учитывается и считается что кодировка текста в файле соответствует BOM (UTF-8, UTF-16). Если префикса BOM в файле нет, то используется текущая кодовая страница ОС.

Если включить галку "**Мониторинг изменений в файле**", то как только содержимое текстового файла изменится (вследствие редактирования или записи нового файла) эти изменения сразу же будут отражены в бегущей строке, которая выведена в эфир.

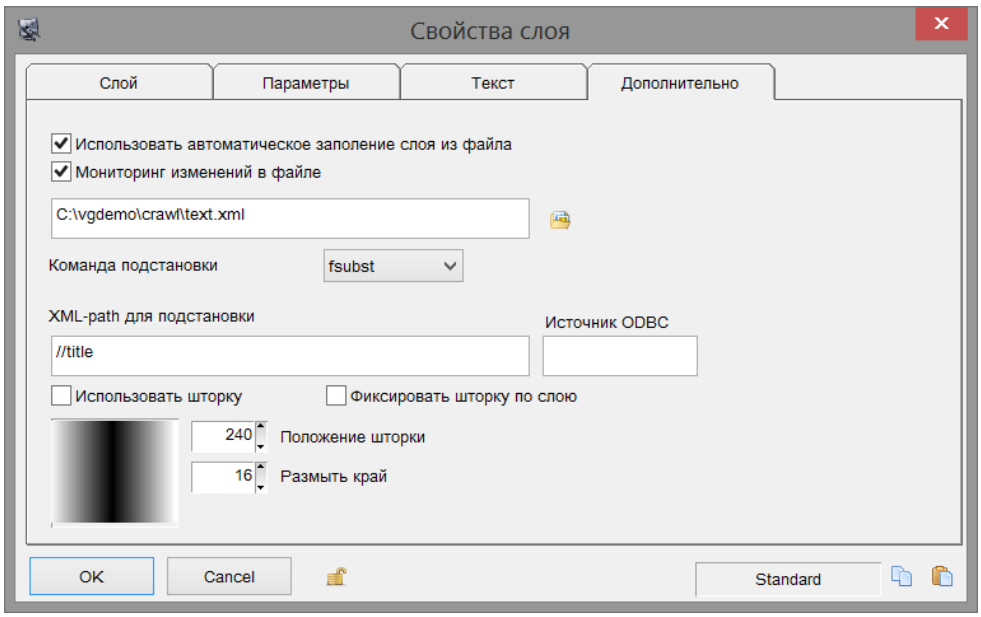
Значения из XML (crawl4.tpl)

Пусть необходимо выполнять заполнение шаблона не просто из текстового файла, а из структурированного хранилища, например XML. Пусть есть файл с такой структурой:

```
<?xml version="1.0" encoding="utf-8"?>
<root>
    <title>Про лисицу</title>
    <text>Быстрый рыжий лис перепрыгнул через спину ленивой собаки</text>
</root>
```

Дополняем наш шаблон двумя слоями, слой **bg_title** -- подложка для заголовка и слой **title** -- текст заголовка. Нужно, чтобы текст из тега **<title>** попал в слой **title**, и текст из тега **<text>** попал в слой **text**.

На закладке "**Дополнительно**" свойств слоя **title** задаем



В качестве имени файла для подстановки указываем **text.xml** и (в отличие от простого текстового файла) задаем X-path для тега **<title>** в поле "**XML-path для подстановки**". Значение пути "**//title**" эквивалентно "**/root/title**"

Для слоя text задаем путь "**//text**".

Получаем шаблон для вывода бегущей строки и заголовка из файла в формате XML.

Барабан (вертикальный ролик)

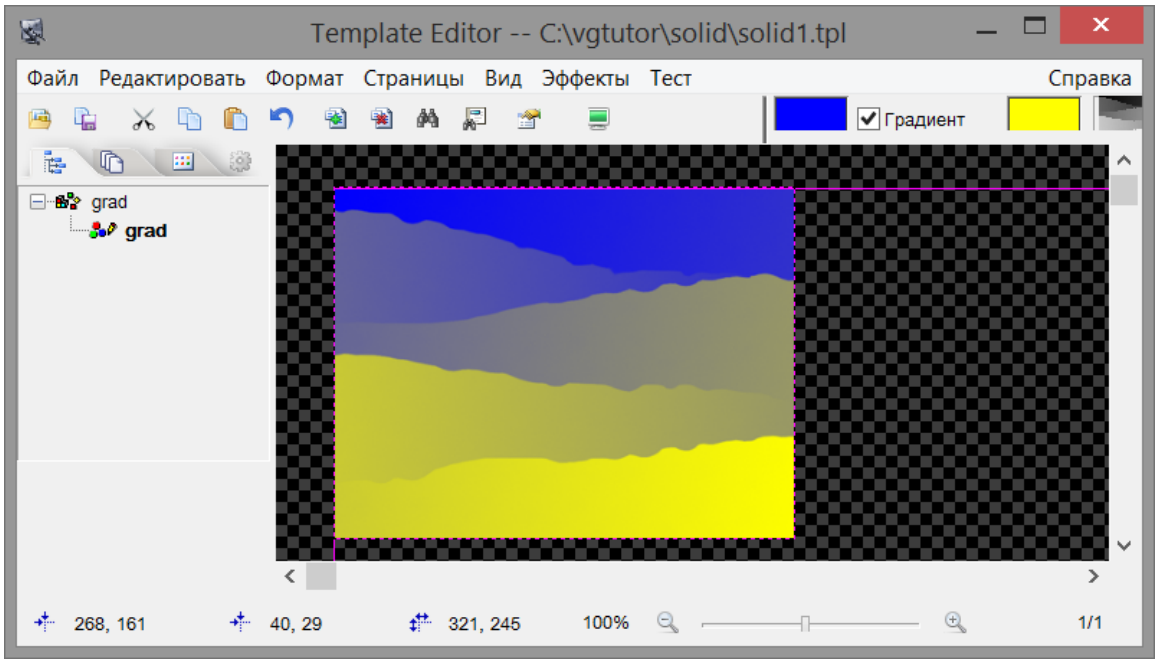
Отличается от бегущей строки командой эффекта -- вместо crawl нужно указывать roll -- и направлением шторки (если барабан с мягким краем).

Цветовые заливки и градиенты c:\vgtutor\solid

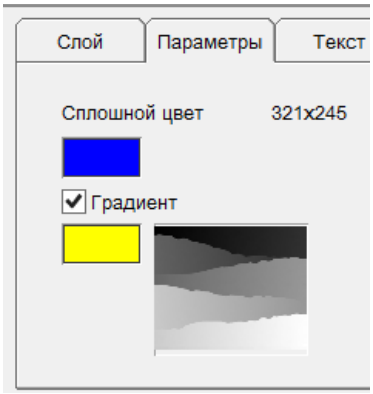
Цветовая заливка является, наверное, самым простым слоем в шаблонах **vgcast**. Но, тем не менее, для слоев этого типа есть несколько специфических команд и приемов работы.

Работа с градиентами (solid1.tpl)

Создаем простой шаблон с одним единственным слоем типа **Solid Fill** с именем **grad**.



В свойствах этого слоя на закладке "**Параметры**" задаем:



Указываем первый (основной) цвет заливки (синий).

Ставим галку "**Градиент**" и задаем второй (дополнительный) цвет заливки (желтый).

Выбираем файл, который будет использоваться как образец для градиента. Это монохромная картинка (если указать цветную, то из нее будет использоваться только яркостная составляющая). Есть смысл для градиентов использовать те же файлы, что и для шторок.

Большой набор шторок можно установить из инсталляционного файла **vgwipe.exe**.

Для нашего примера выбираем файл с именем **BRUSHER.png**

Когда для заливки используется градиент, то черному цвету в образце градиента соответствует основной цвет (в нашем случае синий), а белому -- дополнительный (в примере -- желтый).

Не забываем, что в элементах интерфейса для выбора цвета в редакторе есть ***четыре*** составляющих для описания цвета -- стандартные цветовые слои RGB и отдельная компонента прозрачности A. Если значение прозрачности равно 0, то какие бы цвета мы не выбирали, заполняемый элемент останется полностью прозрачным.

Эффекты **show** и **hide** делаем простейшими, просто показываем шаблон коротким микшером.

Меняем цвет - подстановки (efx subst)

После того, как слои заливки загружен в движок (выведен в эфир) цветовые параметры этого слоя можно менять командой подстановки -- эффект **subst (F5).**

```
00:00:00:00    subst    @.grad  0xFFFFFF00 0xFF0000FF
00:00:00:20    subst    @.grad  0xFF0000FF 0xFFFFFF00
```

Первый параметр команды подстановки **subst** -- это полное имя слоя (т.е. имя шаблона, за которым через точку следует имя слоя). За именем слоя можно указать одно (простая заливка) или два (градиент) числа, которые задают новые цвета в параметрах слоя. Первым указывается основной цвет, вторым -- дополнительный. Если дополнительный цвет не указан, он остается без изменений.

Для представления цвета удобно использовать шестнадцатеричное представление чисел в формате **0xAARRGGBB**, где AA - значение прозрачности, RR - значение красной составляющей, GG - значение зеленой составляющей, BB - значение синей составляющей. Цвета обязательно нужно указывать с компонентой прозрачности.

Для задания цвета можно также использовать выражения, и тогда цветовые компоненты можно указывать в десятичном виде, используя функцию arb(a,r,g,b). В таком случае эффект **subst** можно записать так:

```
00:00:00:00     subst   @.grad  =argb(255,255,255,0) =argb(255,0,0,255)
00:00:00:20     subst   @.grad  =argb(255,0,0,255) =argb(255,255,255,0)
```

В первой команде основной цвет меняется на желтый ARGB = (255, 255, 255, 0) = 0xFFFFFF00, а дополнительный на синий ARGB = (255, 0, 0, 255) = 0xFF0000FF. Во второй команде эффекта через 20 кадров возвращаем цвета к исходным, т.е. основной опять становится синим, а дополнительный - желтым.

При замене цвета с помощью подстановки изменения выполняются "мгновенно".

Плавное изменение цвета (efx smooth)

Если нужно плавно изменить цвет, то можно использовать команду **solid**. Синтаксис команды

solid *layer dur main_color aux_color*

где

layer
полное имя слоя, т.е. имя шаблона, за которым через точку следует имя слоя.

dur
длительность выполнения команды. Если указано просто число NN -- то длительность в кадрах, если указано в форме **dur=NN**, то NN может задавать время в кадрах или в формате HH:MM:SS:FF.

main_color
основной цвет слоя, формат **color 0xAARRGGBB** или **color=0xAARRGGBB** или **color=argb(a,r,g,b).**

aux_color
вспомогательный цвет слоя, формат **aux 0xAARRGGBB** или **aux=0xAARRGGBB** или **aux=argb(a,r,g,b)**. По историческим причинам можно вместо слова **aux** использовать **fore.**

При выполнении команды цвет(а) слоя плавно меняются за указанное время от текущего значения к указанному. В эффекте **smooth (F6)** задано изменение синего цвета на красный, а желтого на белый

```
00:00:00:00     solid   @.grad  dur=20 color=0xFFFF0000 aux=0xFFFFFFFF
00:00:00:20     solid   @.grad  dur=20 color=0xFF0000FF aux=0xFFFFFF00
```

Использование прозрачности (efx trsp)

Поскольку в задании цвета используется компонента прозрачности, то можно выполнять переход в прозрачность, как в эффекте **trsp (F7)**

```
00:00:00:00     solid   @.grad  dur=10 color=0xFF0000FF aux=0x00FFFF00
00:00:00:20     solid   @.grad  dur=10 color=0xFF0000FF aux=0xFFFFFF00
```

В этом эффекте основной цвет остается без изменений, а вспомогательный цвет градиента становится полностью прозрачным.

Работа с текстовыми слоями c:\vgtutor\text

Начальное содержимое текстового слоя и атрибуты текста (шрифт, размер и пр.) задаются в редакторе шаблона в свойствах этого слоя. Если же необходимо изменять текст динамически, то необходимо воспользоваться командами, применимыми к текстовым слоям.

Замена текста (подстановки) (text1.tpl)

В движке **vgcast** основной командой замены содержимого слоев является команды подстановки **subst**. Она применима практически к любым слоям, хотя ее смысл (семантика) меняется в зависимости от типа слоя. Для текстовых слоев синтаксис команды такой:

subst *layer* *text*

где

layer

полное имя слоя, т.е. имя шаблона, за которым через точку следует имя слоя.

text

любая последовательность знаков.

После выполнения команды подстановки содержимое текстового слоя меняется на новый текст, указанный в команде. При этом все атрибуты форматирования текста остаются неизменными.

В тексте подстановки есть два символа, которые используются для дополнительного внутритекстового форматирования, это символы одиночных полиграфических кавычек ‹*format*›. Это символы с кодами U+2039 и U+203A (в юникоде). Все, что заключено между такими символами трактуется по-особенному.

Опишем эффект **show** таким образом, чтобы текст на экране менялся каждые 2 секунды на новый.

```
00:00:00:00   fi      @.*
00:00:00:00   subst   @.text  外国語の学習と教授
00:00:02:00   subst   @.text  Language Learning and Teaching
00:00:04:00   subst   @.text  Изучение и обучение иностранных языков
00:00:06:00   subst   @.text  Tere Daaheng Aneng Karimah
00:00:08:00   subst   @.text  語文教學・语文教学
00:00:10:00   subst   @.text  Enseñanza y estudio de idiomas
00:00:12:00   subst   @.text  ქართული ენის შესწავლა და სწავლება
```

В этом примере нужно обратить внимание на использование большого количества языков в одном слое. Чтобы выводимый текст отображался правильно, нужно выбрать шрифт, который содержит знаки всех алфавитов. В примере используется **Arial Unicode MS**.

Посимвольный вывод текста (text2.tpl)

Команда **type** движка позволяет заменять текст в слое не весь сразу, а посимвольно, имитируя эффект печати на пишущей машинке. Синтаксис практически такой же, но первым параметром команды нужно указать длительность выполнения эффекта, т.е. сколько времени будет "печататься" новый текст.

```
00:00:00:00   type    @.text  dur=25 外国語の学習と教授
00:00:02:00   type    @.text  dur=25 Language Learning and Teaching
00:00:04:00   type    @.text  dur=25 Изучение и обучение иностранных языков
00:00:06:00   type    @.text  dur=25 Tere Daaheng Aneng Karimah
```

При задании параметра скорости или времени выполнения эффекта нужно учитывать, что быстрее, чем один знак за кадр текст выводиться не будет. Т.е. даже если указано, что длительность команды 25 кадров, а в тексте 37 знаков, то выводиться текст будет за 37 кадров.

Вместо длительности команды можно задавать скорость вывода знаков текста

```
00:00:00:00   type    @.text  speed=5 外国語の学習と教授
```

Такой синтаксис команды указывает, что между выводом знаков будет проходить 5 кадров, т.е. для системы PAL скорость печати будет 5 знаков в секунду.

Динамические числовые поля (text3.tpl)

Часто бывает необходимо получить **изменяющиеся** значения чисел, от текущего значения до нового. Для этого в движке есть команд **text**, которая позволяет интерполировать некоторые атрибуты текстовых слоев.

На самом деле наверное есть смысл вставить изменение этих параметров в стандартные опорные точки, но как то исторически сложилось...

text **tpl.layer** ***dur*** ***value*** ***format*** ***color***

где

dur

длительность эффекта в кадрах, может задаваться числом **NN** или фразой **dur=NN**, **dur=HH:MM:SS:FF**

value

новое значение текстового слоя, к которому будет интерполировано текущее значение, может задаваться как **key=NN**, **val=NN**, **value=NN**. **NN** может быть произвольным числом с плавающей точкой или выражением. Для совместимости с предыдущими версиями допускается записывать этот параметр без знака равенства.

format

необязательный параметр, который указывает, как будут отформатированы интерполированные значения. Синтаксис **fmt=*string***., где ***string*** -- произвольная последовательность литер, задающая форматирование числа. В строке допускаются любые элементы форматирования, используемые в языке программирования C. Собственно форматирующие последовательности начинаются со знака **%**. Если строка формата не указана, то используется формат по умолчанию **"%.0f"** -- выводится только целая часть числа. Если в строке формата встречается знаки пробелов или табуляций, нужно заключить всю строку в двойные кавычки. Примеры строк форматирования:

	"rating %.0%%"	"rating %02.0f%%"	"rating %4.1f%%"	"rating %04.1f%%"
2	rating 2%	rating 02%	rating 2.0%	rating 02.0%
2.7	rating 2%	rating 02%	rating 2.7%	rating 02.7%
12.3	rating 12%	rating 12%	rating 12.3%	rating 12.3%
128	rating 128%	rating 128%	rating 128.0%	rating 123.0%

Для совместимости с ранними версиями форматом считается любая строка, которая не начинается с ключевых слов **key**, **val**, **value** и **color**.

color

цвет литер текста. Задается как **color=0xAARRGGBB** или **color=argb(a,r,g,b)**. Цвет текста плавно меняется от текущего к новому.

Итак, создаем шаблон с текстовыми полями **text** и **num**. В поле **num** будем менять численное значение от 10 до 67 за 1 секунду. Эффект **show** описывается как

```
00:00:00:00    text    @.num  dur=1 val=10
00:00:00:00    fi      @.*    10
00:00:00:20    text    @.num  dur=25 val=67
```

В первой команде задаем стартовое значение 10. Для динамических численных полей стартовое значение нужно задавать обязательно. Во время выполнения этой команды слой еще невидимый, поэтому подстановку делаем за 1 кадр. Второй командой показываем все слои микшером за 10 кадров. И в третьей команде указываем, что значение поля **num** нужно изменить от текущего (10) до нового (67) за 25 кадров (1 секунда).

В эффекте **hide** динамически уменьшаем число до 0, тоже за 1 секунду.

```
00:00:00:00    text    @.num  dur=25 val=0
00:00:00:15    fo      @.*    10
```

```
00:00:01:00     del     @
```

Изменяющийся цвет текста (text4.tpl)

Если нужно менять цвет текста, то в команде **text** нужно просто задать еще один параметр -- **color**. В примере одновременно с изменением значения текста от 9.3 до 21.7 будем менять цвет от текущего (желтого) до красного. Форматировать число будем с одним знаком после точки и завершающим символом процента.

```
00:00:00:00     text    @.num  dur=1 val=9.3 fmt="%2.1f%%"
00:00:00:00     fi      @.*    10
00:00:00:20     text    @.num  dur=25 color=0xFFFF0000 val=21.7
00:00:00:20     text    @.text dur=25 color=0xFFFF0000
```

В первой команде задаем начальное значение числа (10) и формат. В дальнейшем формат можно не указывать, будет использоваться предыдущий.

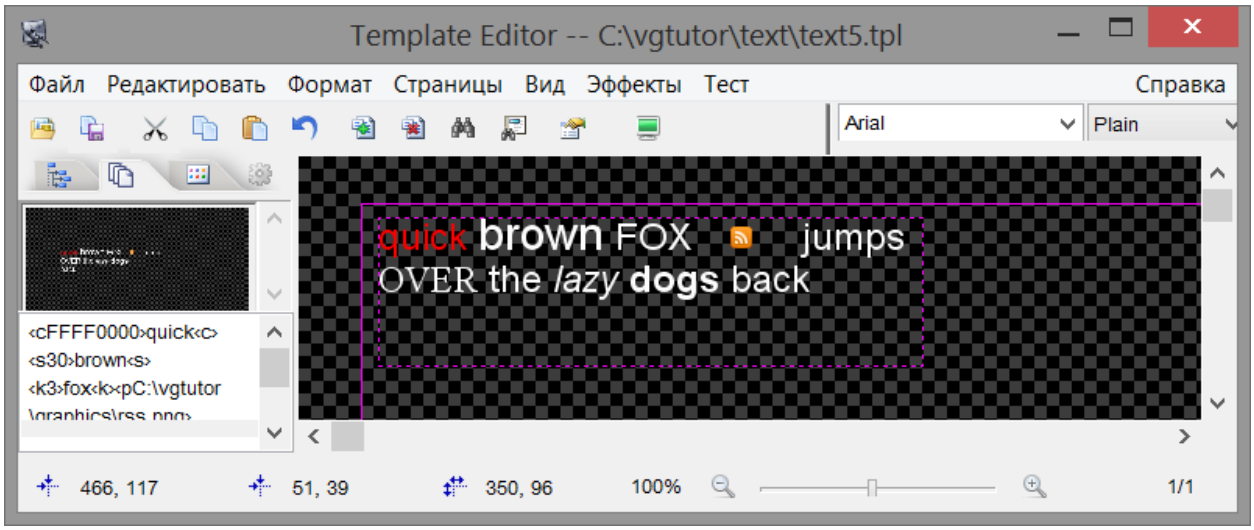
В эффекте **hide** цвет меняем от текущего (красного) к желтому.

```
00:00:00:00     text    @.num  dur=25 val=0 color=0xFFFFFF00
00:00:00:00     text    @.text dur=25 color=0xFFFFFF00
00:00:00:15     fo      @.*    10
00:00:01:10     del     @
```

В этом примере используем не целые числа, а числа с плавающей точкой.

Внутритекстовое форматирование (text5.tpl)

Когда в свойствах текстового слоя задаются параметры форматирования текста -- шрифт, цвет, размер и пр., то эти параметры применяются ко всему тексту в слое. В тех случаях, когда необходимо изменить параметр только для нескольких символов в тексте (например, выделить одно слово жирным шрифтом, или другим цветом) нужно применять форматирование на закладке "Текст" в свойствах слоя.



После дополнительной разметки сам текст становится практически нечитаемым, поскольку он щедро "пересыпан" специальными символами форматирования, но к счастью, такое форматирование используется не так уже и часто. Текст, используемый в примере, приобретает вот такой вид:

‹cFFFF0000›quick‹c› ‹s30›brown‹s› ‹k3›fox‹k›‹pC:\vgtutor\graphics\rss.png› jumps ‹fTimes›OVER‹f› the ‹t2›lazy‹t› ‹t1›dogs‹t› back

Красным цветом выделены специальные последовательности.

Данные из внешних файлов c:\vgtutor\feedtext

Специальный слой типа **Data Feed** используется для подстановки значений в слои **vgcast** из внешних файлов, причем файлы могут располагаться как на локальных носителях, так и на удаленных ресурсах доступных по протоколам FTP или HTTP.

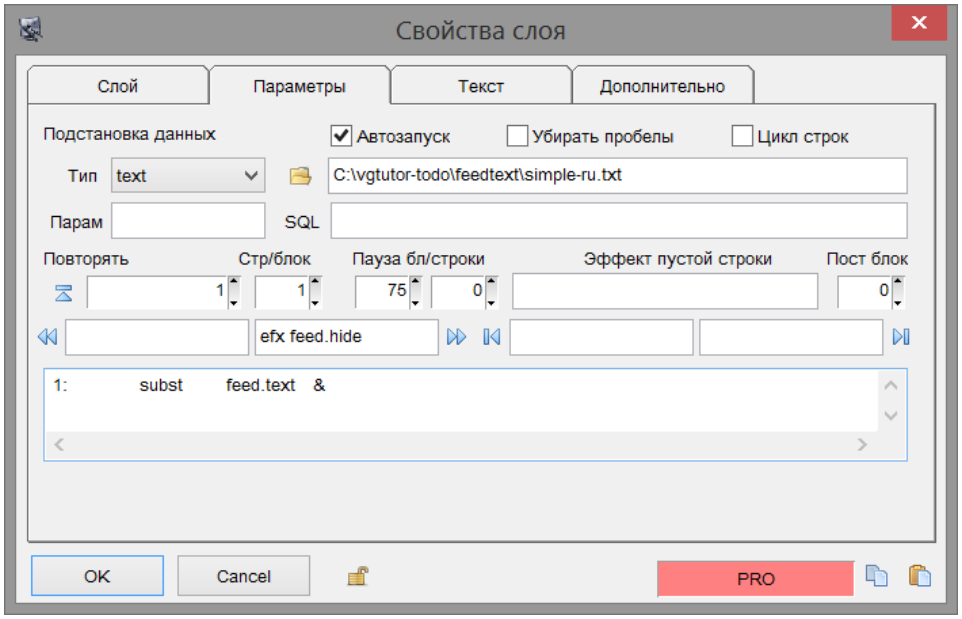
Простая подстановка строк из текстового файла (feed1.tpl)

Пусть есть текстовый файл **simple-ru.txt** с таким содержимым

В горах мое сердце... Доныне я там.
По следу оленя лечу по скалам.
Гоню я оленя, пугаю козу.
В горах мое сердце, а сам я внизу.
Прощай, моя родина! Север, прощай, -
Отечество славы и доблести край.
По белому свету судьбою гоним,
Навеки останусь я сыном твоим!

Необходимо построчно выводить содержимое файла в текстовый слой.

Создаем шаблон с двумя слоями, слой **bg** -- подложка, слой **text** -- слой с текстом. Создаем новый слой с именем **feed**, в свойствах слоя указываем тип слоя **Data Feed**. Переходим на закладку **Параметры**.



Этот слой определяет источник данных и правила подстановки данных в слои. В поле **Тип** выбираем значение **text**, поскольку подставляем данные из простого текстового файла. Давим кнопку **Открыть файл** и выбираем имя файла, который будет источником данных.

Для простых текстовых файлов поля **Парам** и **SQL** оставляем пустыми. Ставим галку **Автозапуск**, это означает, что сразу после загрузки шаблона начнется выборка данных из файла. Выборка строк из файла производится блоками из нескольких строк, в нашем случае по одной строке в блоке (параметр **Стр/блок**). Если хотим показать файл только один раз, оставляем в поле **Повторять** значение **1**. Параметр **Пауза бл/строки** задает (в кадрах) сколько времени будет проходить между выборками следующего блока (или строки), поскольку в нашем случае в блоке содержится ровно одна строка, то достаточно задать только паузу блока **75**, т.е. **3** секунды. В параметрах ⏮ (Эффект начала файла) и ⏭ (Эффект конца файла) можно задать команды **vgcast**, которые выполнятся перед первым чтением данных из файла и, соответственно, после последнего чтения из файла. Если указать в эффекте конца файла команду **efx feed.hide**, то после вывода всех строк из файла шаблон будет удален. С помощью этих параметров мы описали поведение шаблона, откуда и с каким интервалом он выбирает данные.

Осталось описать, куда выполняется подстановка данных. В окне списка команд можно указывать любые команды vgcast. Каждая команда должна начинаться префиксом **NN:** (номер команды, за которым следует знак двоеточия).

```
1:  subst  feed.text  &
```

В теле команды можно использовать метасимволы, в нашем случае это знак **&**. Во время исполнения команды этот знак заменится на текущую выбранную строку текста, т.е. в процессе чтения данных с интервалом в 3 секунды будут выполняться команды

```
subst  feed.text  В горах мое сердце... Доныне я там.
пауза в 3 секунды
subst  feed.text  По следу оленя лечу по скалам.
пауза в 3 секунды
...  ...  ...
```

Эффекты **show** и **hide** можно оставить стандартными, в этом примере просто добавим короткий микшер на 10 кадров.

Подстановка строк из текстового файла блоками (feed2.tpl)

Если необходимо выводить строки из файла не по одной, а блоками по нескольку строк (в примере -- по две строки), шаблон нужно несколько изменить. Вместо одного текстового слоя создаем два, с именами **text[1]** и **text[2]**. Использование квадратных скобок необязательно, важно, чтобы в имени слоя присутствовали числа 1 и 2 -- это номера строк в блоке.

В свойствах слоя **feed** задаем параметр **Стр/блок** равный **2**, т.е. в каждом блоке будет ровно по две строки. Слева от поля Повторять есть пиктограмма ⌛ (Повторять бесконечно), при нажатии мышкой на эту кнопку в поле количества повторений запишется число **2 147 483 647** -- это максимальное положительное целое число. Поскольку данные будут читаться из файла в бесконечном цикле, то эффект конца файла можно не задавать -- он все равно никогда не выполнится.

В окне списка команд задаем

```
1:      subst    feed.text[#]      &
```

В этой команде используется еще один метасимвол -- знак **#**. В процессе выборки строк из файла вместо этого метасимвола в команде будет использоваться номер строки внутри блока, т.е. в нашем примере будут выполняться команды

```
subst  feed.text[1]  В горах мое сердце... Доныне я там.
subst  feed.text[2]  По следу оленя лечу по скалам.
пауза в 3 секунды
subst  feed.text[1]  Гоню я оленя, пугаю козу..
subst  feed.text[2]  В горах мое сердце, а сам я внизу.
пауза в 3 секунды
...  ...  ...
```

Достаточно загрузить этот шаблон в редактор, нажать **F3** (эффект **show**) -- и стишок будет выводиться по две строчки до тех пор, пока не нажмем **F4** (эффект **hide**).

Подстановка из текстового файла в формате CSV (feed3.tpl)

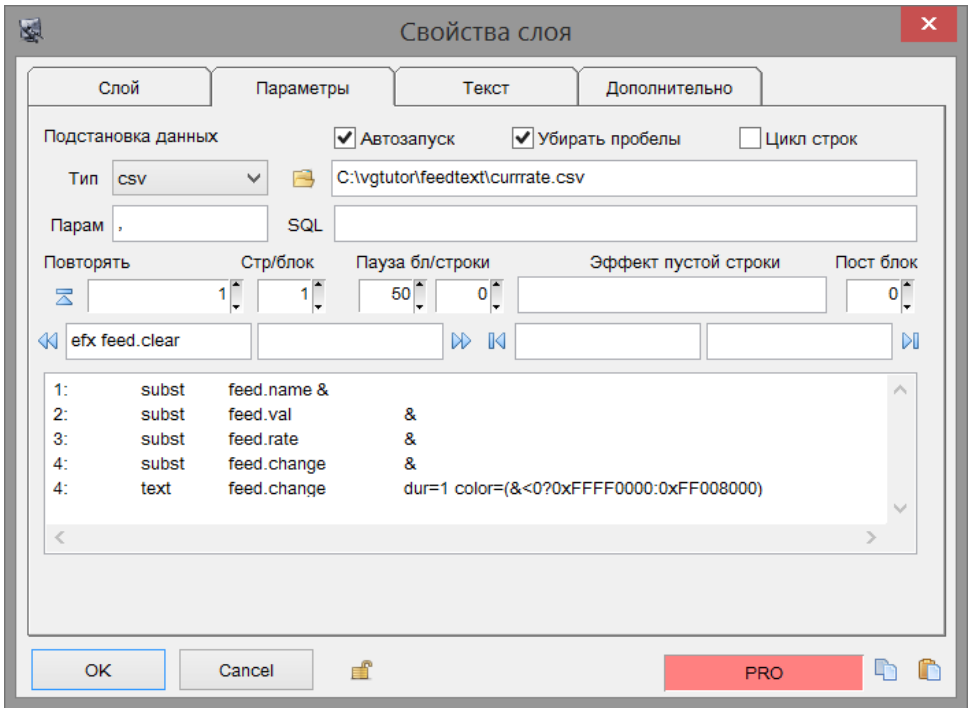
Кроме неструктурированных текстовых файлом можно использовать файлы в формате **CSV** (**C**omma **S**eparated **V**alues -- значения через запятую). В таких файлах строка разбивается на поля, которые отделены друг от друга разделителями (обычно через запятую, но могут использоваться и другие знаки).

В качестве примера рассмотрим текстовый файл **currrate.csv** -- курс некоторой валюты относительно списка других валют.

```
CZK, 100, 64.7807, 1.0529
UZS, 100, 0.5817, 0.0116
HUF, 1000, 57.6125, -1.195
```

SGD, 100, 1093.2827, -24.2737
PLN, 100, 431.5046, 8.377

Все поля в строках отделены друг от друга запятыми. Первое поле -- это код валюты (слой **name**), второе -- количество (слой **val**), третье -- курс обмена (слой **rate**), четвертое -- изменение курса (слой **change**). В зависимости от изменения курса нужно раскрашивать его в **красный** (для отрицательных значений) или **зеленый** (для положительных) цвет. Параметры слоя **feed** для файла CSV имеют такой вид



Тип файла, естественно, выбран как **csv**, и поле **Парам** указан знак-разделитель отдельных элементов строки, в нашем случае -- знак запятой. Обратите внимание на галку **Убирать пробелы** -- после разбивание строки на поля в каждом поле убираются ведущие и завершающие пробелы, табуляторы и знаки конца строки.

В поле команд номера в начале строк соответствуют порядковым номерам полей в строке, так первое поле -- это код валюты, а четвертое -- изменение курса. Для поля изменения курса выполняется две команды, поэтому последние две строки имеют одинаковые номера. Предпоследняя команда выполняет подстановку значения в слой, а последняя команда задает цвет текстового слоя, в зависимости от знака поля **change**:

```
4:    text    feed.change    dur=1 color=(&<0?0xFFFF0000:0xFF008000)
```

Или в альтернативном виде

```
4:    text    feed.change    dur=1 color=if(&<0,0xFFFF0000,0xFF008000)
```

Подробно команда **text** рассматривается в разделе **Работа с текстовыми слоями**, но в этом примере используется выражение для вычисления цвета текста. Смысл этого выражения такой -- если значение четвертого поля в строке меньше нуля, то выбирается красный цвет (0xFFFF0000), если больше (или равно) нулю -- зеленый (0xFF008000) цвет. Можно еще больше усложнить условие, например если изменение курса равно нулю -- вообще не выводить это поле (задать полностью прозрачный цвет текста):

```
4:    text    feed.change    dur=1 color=if(&<0,0xFFFF0000,if(&>0,0xFF008000,0x00000000))
```

В результате получаем такую последовательность в эфире:

Данные из внешних файлов XML c:\vgtutor\feedxml

Специальный слой типа **Data Feed** используется для подстановки значений в слои **vgcast** из внешних файлов, причем файлы могут располагаться как на локальных носителях, так и на удаленных ресурсах доступных по протоколам FTP или HTTP. В тех случаях, когда информация структурирована, удобно использовать файлы в формате XML. Эти файлы легко формировать из приложений, многие онлайн-ресурсы предоставляют доступ к данным в этом формате (например, потоки RSS для новостей).

Простая подстановка из файла (xml1.tpl)

Пусть есть файл в формате XML **currrate.xml** с такой структурой

```
<?xml version="1.0" encoding="windows-1251"?>
<chapter url="http://bank-ua.com/export/currrate.xml">
 <item>
  <date>2014-08-26</date>
  <code>203</code>
  <char3>CZK</char3>
  <size>100</size>
  <name>чеських крон</name>
  <rate>64.7807</rate>
  <change>1.0529</change>
 </item>
... ... ...
 <item>
  <date>2014-08-26</date>
  <code>860</code>
  <char3>UZS</char3>
  <size>100</size>
  <name>узбецьких сумів</name>
  <rate>0.5817</rate>
  <change>0.0116</change>
 </item>
</chapter>
```

Необходимо выводить содержимое файла в текстовые слои.

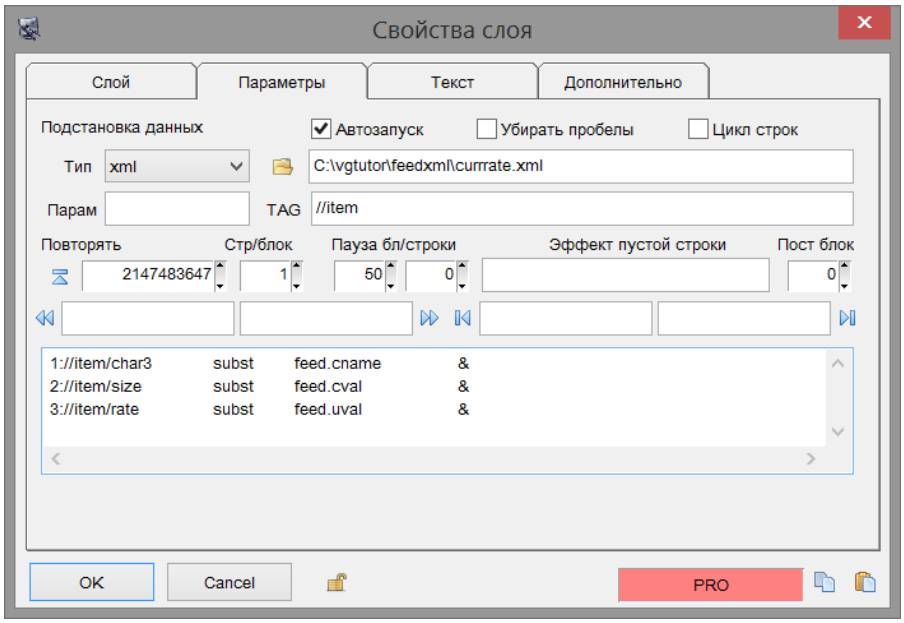
Создаем шаблон с двумя слоями, слой **bg** -- подложка, слой **cname** -- слой для кода валюты, тег **<char3>**, **cval** -- слой для тега **<size>**, слой **uval** -- для текущего курса, тег **<rate>**. Слой с именем **uah** используется для имени валюты, курс которой сформирован в XML-файле.

В отличие от текстовых файлов данных (или файлов в формате CSV) структура XML более гибкая. В нашем случае, информация по курсу каждой валюты организована повторяющимися группами внутри тегов **<item>**.

Создаем новый слой с именем **feed**, в свойствах слоя указываем тип слоя **Data Feed**. Переходим на закладку **Параметры**. Этот слой определяет источник данных и правила подстановки данных в слои. В поле **Тип** выбираем значение **xml**, поскольку подставляем данные из файла в формате XML. Давим кнопку **Открыть файл** и выбираем имя файла, который будет источником данных.

Ставим галку **Автозапуск**, это означает, что сразу после загрузки шаблона начнется выборка данных из файла.

Для доступа к тегам файла XML используется синтаксис X-Path (точнее, некоторое подмножество спецификации X-Path). Так, если мы хотим сослаться на тег **<item>**, то нужно указать путь **//item** (или **/chapter/item**). если хотим сослаться на тег **<char3>**, то указываем путь **//item/char3** и т.д.



В отличие от простых текстовых файлов в поле **TAG** нужно указать тот тег, который образует повторяющиеся группы данных. В нашем примере это теги **<item>**, поэтому в поле **TAG** указываем "путь" к этому тегу.

Выборка строк из файла производится блоками из нескольких строк, в нашем случае по одной строке в блоке (параметр **Стр/блок**). Если хотим показать файл только один раз, оставляем в поле **Повторять** значение **1**, если хотим крутить файл бесконечно -- давим на кнопку (Повторять бесконечно). Параметр **Пауза бл/строки** задает (в кадрах) сколько времени будет проходить между выборками следующего блока (или строки), поскольку в нашем случае в блоке содержится ровно одна строка, то достаточно задать только паузу блока **50**, т.е. **2** секунды. С помощью этих параметров мы описали поведение шаблона, откуда и с каким интервалом он выбирает данные.

Осталось описать, куда выполняется подстановка данных. В окне списка команд можно указывать любые команды **vgcast**. Каждая команда должна начинаться префиксом **NN:** (номер команды, за которым следует знак двоеточия), и в отличие от текстовых файлов, нужно указать путь к тому тегу файла, из которого будут выбираться данные для подстановки:.

```
1://item/char3  subst  feed.cname  &
2://item/size   subst  feed.cval   &
3://item/rate   subst  feed.uval   &
```

В теле команды можно использовать метасимволы, в нашем случае это знак **&**. Во время исполнения команды этот знак заменится на содержимое тега, указанного после номера команды.

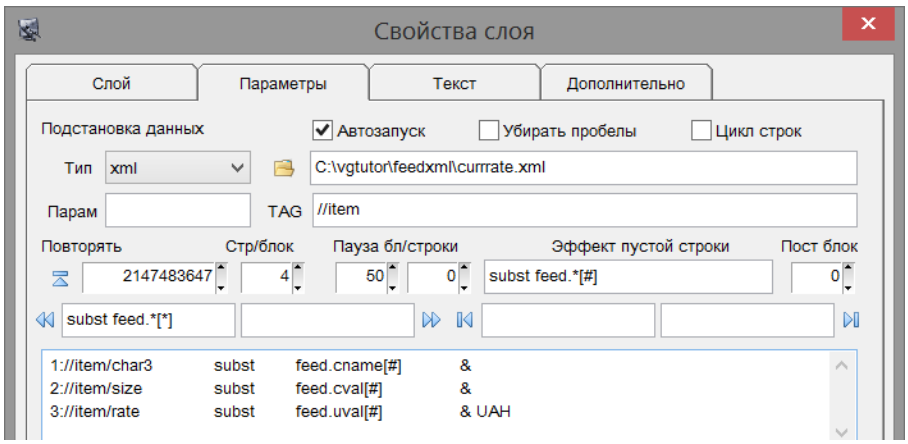
Эффекты **show** и **hide** можно оставить стандартными, в этом примере просто добавим короткий микшер на 10 кадров.

Подстановка из файла XML блоками (xml2.tpl)

Если необходимо выводить строки из файла не по одной, а блоками по нескольку строк (в примере -- по четыре строки), шаблон нужно несколько изменить. Вместо одного текстового слоя создаем четыре, с именами **cname[1]**, **cname[2]** т.д. Использование квадратных скобок необязательно, важно, чтобы в имени слоя присутствовали числа 1, 2, 3, 4 -- это номера строк в блоке.

В свойствах слоя **feed** задаем параметр **Стр/блок** равный **4**, т.е. в каждом блоке будет ровно по четыре строки. Слева от поля Повторять есть пиктограмма (Повторять бесконечно), при нажатии мышкой на эту кнопку в поле количества повторений запишется число **2 147 483 647** -- это максимальное положительное целое число. Поскольку данные будут читаться из файла в бесконечном цикле, то эффект конца файла можно не задавать -- он все равно никогда не выполнится.

В этом шаблоне используется еще один метасимвол -- знак **#**. В процессе выборки строк из файла вместо этого метасимвола в команде будет использоваться номер строки внутри блока, т.е. в нашем примере будут выполняться команды.



В эффекте начала файла задаем подстановку во все слои шаблона, имена которых совпадают с маской **feed.*[*]**, в нашем шаблона это все слои, **кроме** слоя **bg**. Команда подстановки **subst feed.*[*]** подставит пустые строки во все текстовые слои перед началом выборки данных. Такая команда нужна, чтобы не показывать то содержимое текстовых слоев, которое было задано в редакторе.

В файле с курсами валют есть 26 групп **<item>**, т.е. информация по 26 валютам. Поскольку мы выдаем информацию группами по 4 валюты, в последней группе (блоке) нужно выдавать только 2 валюты, т.е. для двух последних строк блока с индексами 3 и 4 нужно стереть содержимое текстовых слоев с этими индексами. Это действие задается в поле **Эффект пустой строки**, эта команда выполнится для тех групп блока, для которых "не хватило" информации в файле, т.е. в последнем блоке файла кроме команд подстановки данных с индексами 1и 2 будут исполнены еще две команды:

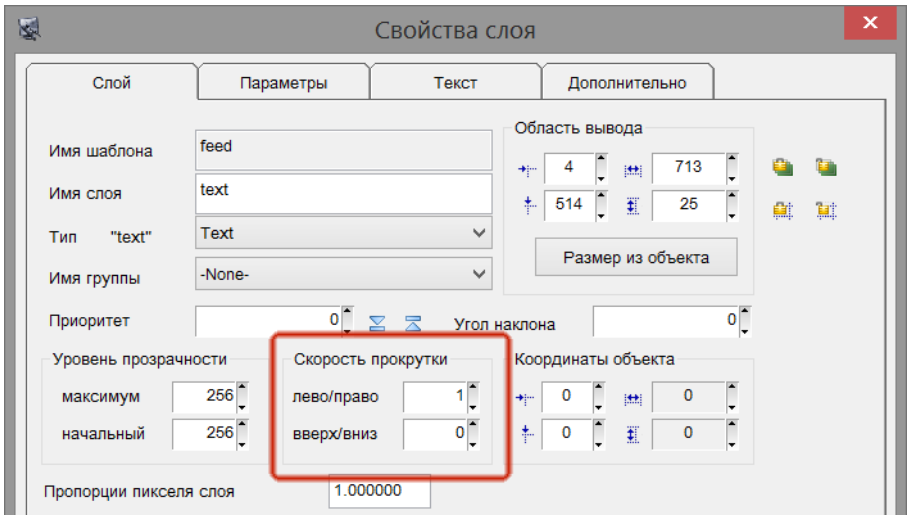
subst feed.*[3]
subst feed.*[4]

Достаточно загрузить этот шаблон в редактор, нажать **F3** (эффект **show**) -- и курсы будет выводиться по четыре строчки до тех пор, пока не нажмем **F4** (эффект **hide**).

Подстановка из файла в бегущую строку (xml3.tpl)

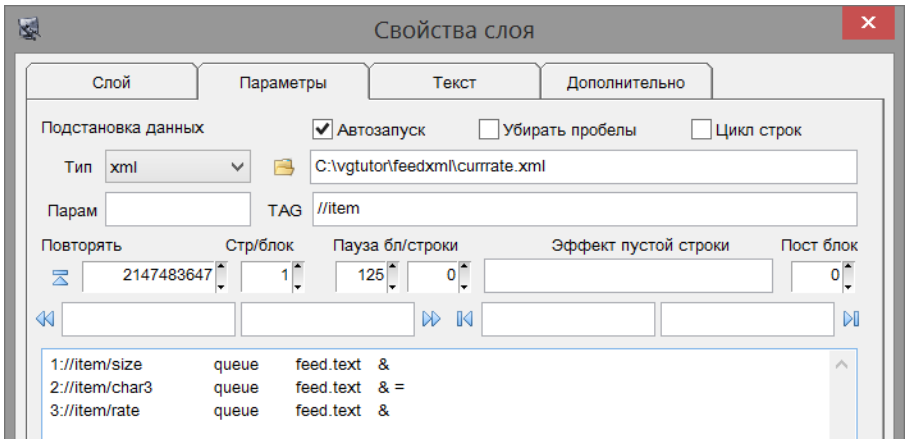
Если необходимо выводить данные курсов валют в виде бегущей строки, применяется техника, отличная от вывода стандартных бегущих строк.

Создаем шаблон со слоем **bg** (задник) и **text** (текст бегущей строки). В свойствах слоя текст на первой закладке **Слой** есть группа параметров **Скорость прокрутки**.



Если в полях **лево/право** или **вверх/вниз** стоят ненулевые значения, то этот слой (его содержимое) все время равномерно движется с указанной скоростью. Если число положительное, слой движется влево (или вверх), если отрицательное -- вправо (или вниз). Для таких слоев есть специальная команда **vgcast** -- **queue** -- поставить в очередь.

Именно эта команда будет использоваться в параметрах слоя **feed** вместо команды подстановки.



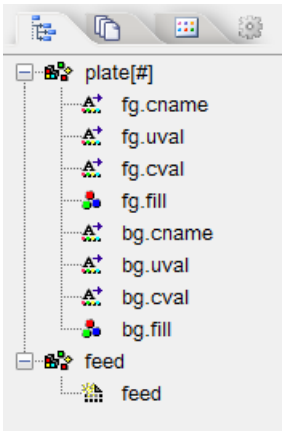
При использовании команды **queue** нужно величину интервала между блоками. Этот интервал должен быть достаточно большим, чтобы между блоками в бегущей строке оставалось некоторое расстояние. Определить этот интервал достаточно нетривиально, поскольку он зависит и от скорости движения слоя, и от количества литер в подставляемых данных (а это величина вообще неизвестная), так что проще всего задавать его "на глаз", запустить шаблон в эфир и оценить время, за которое появляется одна группа данных. В нашем случае этот интервал равен примерно 5 секундам, или 125 кадрам. Вообще говоря, ничего страшного не произойдет и в том случае, если **Пауза бл/строки** окажется короче, чем время появления блока. Но при этом следующие блоки будут становится в очередь раньше, чем пройдет предыдущий блок и со временем очередь будет расти, потребляя оперативную память.

Эффекты **show** и **hide** оставляем стандартными, загружаем шаблон в редактор, давим **F3**, появляется бегущая строка с курсами валют.

Когда нужно остановить вывод строки давим **F4**.

Подстановка из файла XML блоками и клонирование шаблонов (xml4.tpl)

Если вернуться к примеру **xml2.tpl**, то видно, что в нем нужно было описать большое количество слоев, по три слоя для каждой группы данных. И при этом все группы данных имеют одинаковый формат. Если же для эффекта смены данных использовать эффекты со сменой переднего и заднего плана (см. пример **flip**), то количество слоев возрастает еще больше.

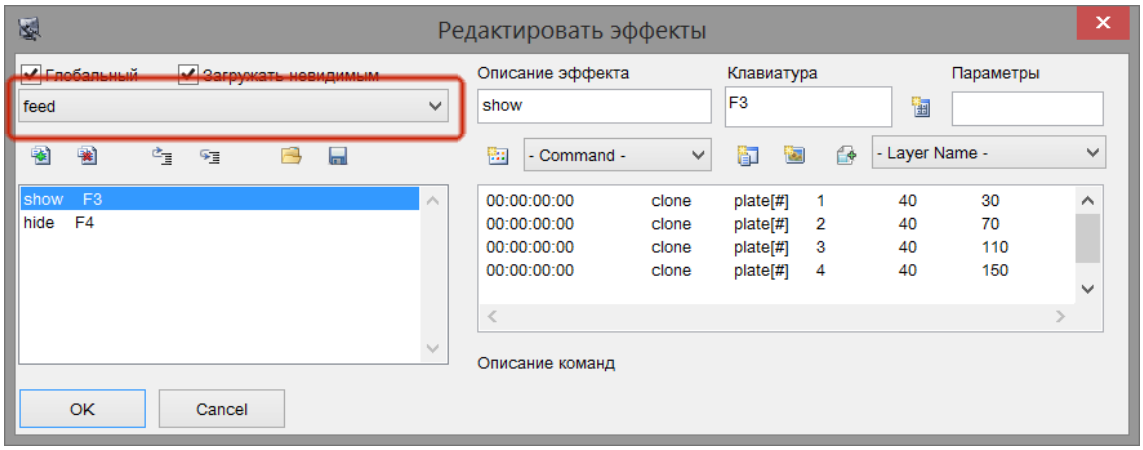


Для создания таких однородных групп слоев можно использовать механизм клонирования шаблонов. Для этого создаем в одном документе два шаблона, один для слоя **feed**, второй -- для описания группы с именем **plate[#].**

Использование знака **#** в имени шаблона указывает, что это **мастер-шаблон**, на основании которого можно создавать произвольное количество **экземпляров** шаблонов с именами **plate[1]**, **plate[2]** и т.д.

Чтобы создать еще один шаблон в том же документе достаточно щелкнуть правой мышкой в поле структуры документа и выбрать команду **Новый шаблон**.

В редакторе эффектов задаются эффекты для каждого шаблона, т.е. можно создавать их и для шаблона **feed** , и для шаблона **plate[#].**

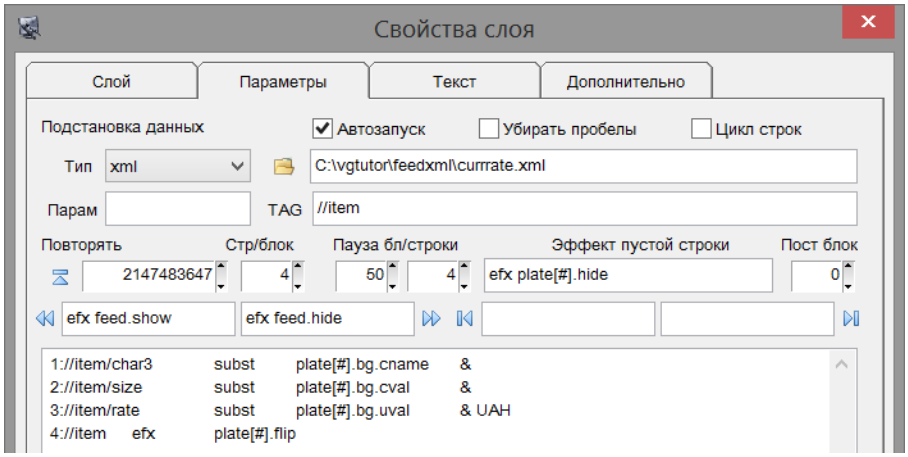


В красной рамке указан комбо-бокс для выбора шаблона. В шаблоне **feed** в эффекте **show** выполняем клонирование шаблона **plate[#]**, делаем 4 клона с нолмерами от1 до 4 и координатами по вертикали от 30 до 150. Горизонтальные координаты одинаковы и равны 40.

Эффект **hide** для шаблонов с клонами тоже несколько специфичен:

```
00:00:00:00    feed    @.feed  stop        останавливаем feed
00:00:00:00    fo      plate[*].*  10      убираем микшером все слои всех шаблонов plate
00:00:00:15    del     plate[*]            удаляем все шаблоны plate
00:00:00:15    del     @                   удаляем шаблон feed
```

Для шаблона **plate[#]** задаем три эффекта -- **show**, **hide** и **flip**. В параметрах слоя **feed** задаем



В общем, все как и в примере xml2.tpl, только индексируются не слои, а шаблоны.

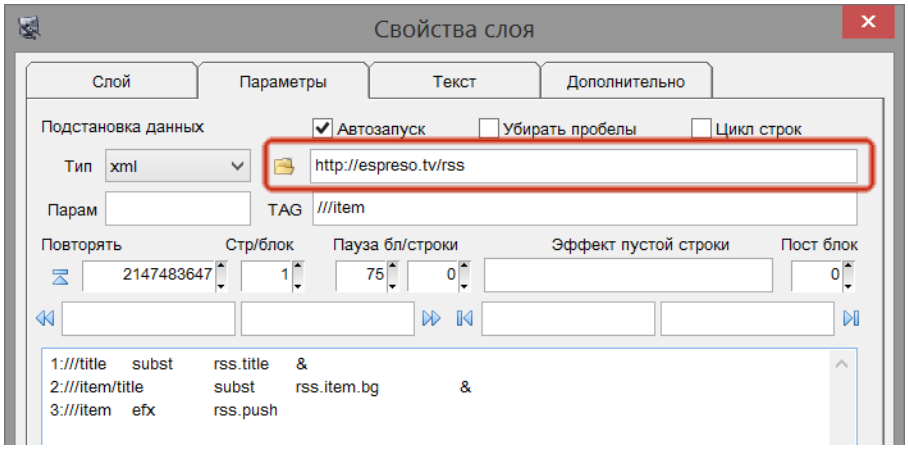
Вывод потоков RSS (rss.tpl)

Поток RSS -- этот тот же файл в формате XML, но он имеет достаточно строгую спецификацию тегов, и обычно используется на WEB-ресурсах для оперативной доставки новостей.

```
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
 <channel>
  <title>заголовок канала</title>
  ...  ...  ...
  <item><title>заголовок новости</title>  ...  </item>
  ...  ...  ...
  <item><title>заголовок новости</title>  ...  </item>
 </channel>
</rss>
```

В принципе, параметры слоя **feed** для RSS-потоков ничем не отличаются от рассмотренных выше.

Чтобы оперативно обновлять информацию в поле источника данных нужно указать ссылку непосредственно на RSS-поток, в нашем случае **http://espreso.tv/rss**. Но для работы в таком режиме требуется доступ к интернету. Если с графической станции нет доступа к интернету, можно на любой другой машине в сети зайти по этой ссылке браузером и сохранить страничку как XML, затем перенести на графическую станцию и указать имя XML-файла.



Для смены заголовков новостей в слое **item** используется эффект "выталкивания" данных снизу-вверх, создаем эффект с именем **push** с такими командами:

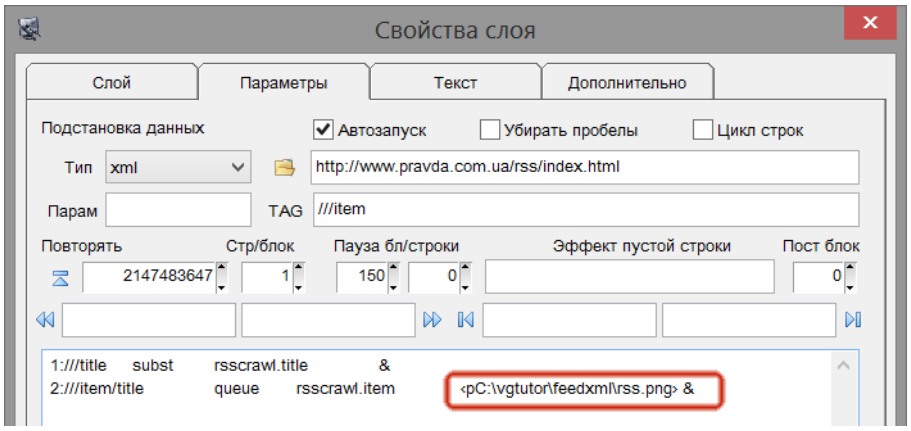
```
00:00:00:00     key     @.item.fg n0 y=0 n25 y=-ht(@.item.fg)
00:00:00:00     key     @.item.bg n0 y=ht(@.item.bg) n25 y=0
00:00:01:01     swap    @.item.fg @.item.bg
```

Первая команда сдвигает видимую часть текста (слой **item.fg**) вверх, одновременно вторая команда сдвигает слой **item.bg** снизу-вверх. После выполнения сдвигом обмениваем имена слоев командой **swap**.

Из всей информации в RSS мы будем использовать только заголовок канала **/rss/channel/title** и заголовки нововстей **/rss/channel/item/title**. При работе со списком команд в параметрах слоя **feed** нужно учитывать тот факт, что команда выполняется при обработке ***закрывающего*** тега. Таким образом, сначала выполняются все (в примере -- одна☺) подстановки в невидимые слои внутри группы **<item>**, и при достижении закрывающего тега **</item>** выполняется эффект смены слоев **push**.

Вывод потоков RSS в бегущую строку (rsscrawl.tpl)

Используем такой же способ вывода данных, как в примере xml3.tpl..



Пожалуй единственная особенность этого примера -- это вставка пиктограммы RSS (графический файл) перед очередным заголовком новости. Ссылка на файл (и вообще все команды внутритекстового форматирования) вставляется как команда форматирования, заключенная в полиграфические угловые скобки ***‹команда›***. Это знаки с кодами U+2039 (0x8B) и U+203A (0x9B), Эти знаки можно вводить с цифровой клавиатуры (в режиме NumLock) с нажатой кнопкой Alt как Alt+0139 и Alt+0155.

Проще вводить команды форматирования текста на закладке Текст в свойства шаблона. Для ввода графики давим на кнопку (**Вставить изображение**), выбираем файл с картинкой, и в текстовом поле формируется нужная

команда. Выбираем все литеры команды, давим Ctrl+C, переходим на закладку Параметры, позиционируем текстовый курсор в нужном месте команды и давим Ctrl+V.

Кэширование файлов данных XML

При работе объектов типа **Data Feed** с файлами в формате XML есть один неприятный момент -- **vcast** открывает файл с данными в начале цикла обработки и держит открытым все время, пока выполняет выборку тегов. Т.е. файл остается заблокированным для записи практически все время. Если же другое приложение хочет обновлять данные в файле динамически, то возникают коллизии доступа к файлу. Однако этих конфликтов чтения-записи можно избежать, если объект **Data Feed** будет работать не с оригинальным файлом, а его копией кэше. В таком режиме объект **Data Feed** в начале каждого цикла чтения файла выполняет копирование оригинала в кэшированную копию, при этом оригинальный файл блокируется на очень короткий промежуток времени.

Для включения режима кэширования нужно в поле Парам в свойствах слоя **Data Feed** указать имя файла-оригинала (например, **currrate_src.xml**), и тогда содержимое этого файла будет копироваться в рабочий файл **currrate.xml** в начале каждого цикла обработки.

Использование атрибутов тегов XML

Кроме значений тегов в файлах XML может понадобиться выбирать еще и атрибуты тегов. Так в спецификации формата RSS может использоваться тег **<enclosure>**, в котором задается ссылка на изображение, связанное с темой. Эта ссылка описывается как атрибут **url**:

```
<enclosure length="8" url="http://image.112.ua/300x225/Nov2013/805.jpg" type="image/jpeg"/>
```

Для выборки таких элементов XML нужно в командах слоя feed использовать такой синтаксис X-Path:

```
2:///item/enclosure/@url        subst    rss.pict  &
```

Нужно заметить, что **vgcast** в качестве ссылки на файл картинки получит ссылку на интернет-ресурс, но он правильно обработает URL с протоколом **http** или **ftp**. Правда выборка картинки будет осуществляться без кэширования, так что нужно иметь надежный и быстрый канал доступа в интернет.

Взаимодействие нескольких шаблонов c:\vgtutor\multi

В большинстве случаев поведение шаблонов в системе **vgcast** не зависит друг от друга. Однако команды движка позволяют использовать изменение параметров не только внутри одного шаблона, но и менять поведение и параметры **других** шаблонов в системе.

Рассмотрим пример, когда один шаблон влияет на поведение другого шаблона.

Два шаблона без связей (multi1.vgd)

Пусть есть два шаблона, одни с именем **tema**, второй – с именем **pgm**. Для удобства представления примера оба шаблона находятся в одном документе редактора как отдельные страницы.

При выводе шаблона **tema** выполняются очень простые действия – он просто появляется на экране некоторым эффектом.

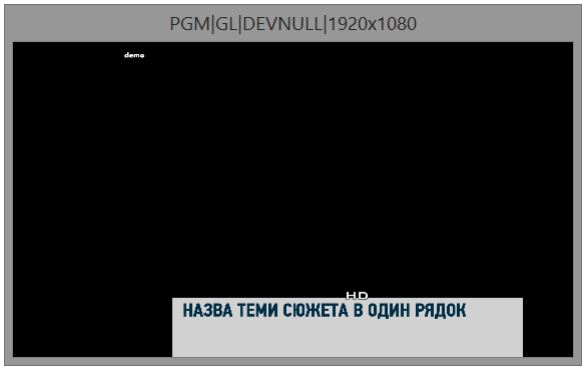


Рисунок 1.

Сам по себе эффект достаточно простой – и подложка (слой **tema.mov**) и текст (слой **tema.text**) выползают из-под нижней части экрана за 6 кадров. Скрипт эффекта **show**:

```
00:00:00:00     key     @.mov   n=0 y=ht(@.mov) t=0 n=6 y=0 t=256
00:00:00:04     key     @.text  n=0 y=ht(@.mov) t=0 n=6 y=0 t=256
```

Аналогично, шаблон **pgm** находится в правой части экрана и перекрывает шаблон **tema**. Т.е., если сначала вывести шаблон **tema**, а потом, не убирая его с экрана вывести шаблон **pgm**, то результирующая картинка станет такой:

Рисунок 2.

Опять таки, скрипт эффекта **show** шаблона **pgm** совершенно прост:

```
00:00:00:00     tkey    @       n=0 x=1920 n=6 x=1320
00:00:00:00     fi      @.*
```

Т.е. весь шаблон **pgm** (включая все его слои) за 6 кадров выдвигается из-за правого края экрана.

Простая связь шаблонов (multi2.vgd)

Однако, хотелось бы, чтобы в таком случае текст в шаблоне **tema** был переформатирован и был полностью виден. Для этого эффект **show** шаблона **pgm** модернизируем так, чтобы в нем были команды управления **другим** шаблоном, в нашем случае, шаблоном **tema**. Фактически, при появлении шаблона **pgm** нужно указать, что размеры слоев шаблона **tema** изменились. Новый скрипт шаблона **pgm** имеет такой вид:

```
00:00:00:00     tkey    @       n=0 x=1920 n=6 x=1320
00:00:00:00     fi      @.*
00:00:00:00     lkey    tema.text       n=6 h=1155-800
00:00:00:00     lkey    tema.mov        n=6 h=1205-800
```

Обратить внимание – для слоев шаблона **pgm** (текущего шаблона) в имени слоя в качестве имени шаблона используется знак **@** -- имя текущего шаблона. В принципе, можно было бы писать вместо значка **@** слово **pgm**. Для команд, выделенных красным цветом, используется явное имя шаблона, к которому эти команды применяются – шаблон **tema**.

Красным выделены новые команды эффекта **show**. В этих командах указывается, что размер слоя **tema.tex**t будет принимать новое значение 1155-500 за 6 кадров, а размер слоя **tema.mov** –1205-500. Значение 1155 – это исходная ширина слоя **tema.text**, а 1205 – исходная ширина слоя **tema.mov**. Фактически, мы уменьшаем ширину каждого слоя шаблона **tema** на 500 пикселей.

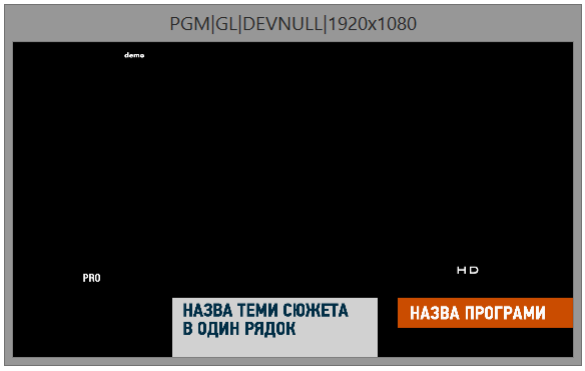


Рисунок 3.

В результате изменения горизонтального размера слоя **tema.text** весь текст этого слоя будет переформатирован в соответствие с параметрами формата (в нашем примере – будет произведен перенос текста на вторую строку).

Далее, хотелось бы, чтобы при уходе шаблона **pgm** из эфира, шаблон **tema** вернулся к своему исходному виду (ну, конечно, если это необходимо☺). Для это эффект **hide** шаблона **pgm** должен опять поменять параметры шаблона **tema** – на самом деле просто вернуть размеры слоев в их исходное состояние.

```
00:00:00:00     tkey    @       n=0 x=1320 n=6 x=1920
00:00:00:04     lkey    tema.text       n=6 h=1155
00:00:00:04     lkey    tema.mov        n=6 h=1205
00:00:00:14     del     @
```

Эти команды выделены красным в примере. Т.е. если теперь выполнить для шаблона **pgm** эффект **hide**, то картинка вернется в состояние ***Рисунок 1***.

Связанные шаблоны с переменными (multi3.vgd)

Однако есть еще одна проблема. Рассмотрим ситуацию, когда ***сначала*** выводится шаблон **pgm**, а затем – шаблон **tema**. В таком случае опять получим картинку, как на ***рисунке 2***. Это не есть хорошо и с этим нужно бороться.

Для решения проблемы можно использовать аппарат ***переменных*** движка **vgcast**. Т.е. можно объявить переменную, например, с именем **SideOff**, в которую будем записывать текущее значение той величины, на которую нужно уменьшить ширину слоев шаблона **tema**.

Модифицируем скрипт команды **show** для шаблона **tema**:

```
00:00:00:00     lkey    @.text   n=0 h=1155-SideOff
00:00:00:00     lkey    @.mov    n=0 h=1205-SideOff
00:00:00:00     key     @.mov    n=0 y=ht(@.mov) t=0 n=6 y=0 t=256
00:00:00:04     key     @.text   n=0 y=ht(@.mov) t=0 n=6 y=0 t=256
```

Красным выделены новые строки в списке команд эффекта. Если переменная **SideOff** еще не была объявлена, то считается, что ее значение равно **0**.

Аналогично, модифицируем скрипт команды **show** для шаблона **pgm**:

```
00:00:00:00     tkey    @        n=0 x=1920 n=6 x=1320
00:00:00:00     fi      @.*
00:00:00:00     int     SideOff =500
00:00:00:00     lkey    tema.text        n=6 h=1155-SideOff
00:00:00:00     lkey    tema.mov         n=6 h=1205-SideOff
```

Зеленым выделена команда объявления переменной **SideOff**, это переменная типа **int** и ей присваивается значение **500**. Т.е. если сначала будет выполнен эффект **show** шаблона **pgm**, то переменная **SideOff** примет значениет 500, и если затем выполнить эффект **show** шаблона **tema**, то ширина слоев этого шаблона уменьшится на значение переменной **SideShow**, т.е. на 500. Получим картинку как на ***рисунке 3***.

В эффекте **hide** шаблона **pgm** тоже меняем значение переменной **SideOff**

```
00:00:00:00     tkey    @        n=0 x=1320 n=6 x=1920
00:00:00:00     int     SideOff =0
00:00:00:04     lkey    tema.text        n=6 h=1155-SideOff
00:00:00:04     lkey    tema.mov         n=6 h=1205-SideOff
00:00:00:14     del     @
```

Т.е. возвращаем все на круги своя.

Связанные шаблоны с переменными – вариант (multi4.vgd)

Практически тот же набор шаблонов, что и варианте 3, но характеристики текстового слоя другие, вместо переноса слов задан режим масштабирования текста.