

VGCAST – справочник

Легенда	1
Концепция.....	1
Настройки программы	1
Журнал событий	1
Протокол обмена и кодировка текстовых данных.....	2
Именованние объектов	2
Команды управления	2
Команды работы с шаблонами	3
Клонирование шаблонов.....	3
Управление прозрачностью объектов	4
Подстановки	4
Расположение слоев на экране.....	5
Расположение шаблонов на экране.....	6
Команды слоев	7
Работа со звуком.....	7
Работа с объектами 3D.....	7
Информационные команды.....	8
Описание шаблона.....	9
Пример организации связи	17

Легенда

Зеленым цветом выделены новые возможности. **Красным цветом** выделены нереализованные возможности. **Синим цветом** выделены временные решения, которые почти наверняка будут изменены.

Концепция

Исполняющая система **vgcast** (движок) позволяет в реальном времени отображать графическую информацию для оформления эфира. Общая структура отображаемых данных задается шаблонами, в которых описана геометрия данных. В интерпретатор может быть загружено одновременно произвольное количество шаблонов. Шаблоны могут загружаться и удаляться динамически. Управление интерпретатором выполняется с помощью команд, которые задают конкретное поведение элементов шаблонов и позволяют динамически менять данные в шаблонах.

Связь между движком и остальными приложениями выполняется по протоколу TCP/IP, собственно движок является сервером, которому по указанному порту приложения могут отправлять команды. Одновременно может работать сколько угодно приложений (клиентов). Когда команды приложения загружают шаблоны в движок, то фиксируется «владелец» шаблона, т.е. то приложение, которое инициировало создание шаблона. Когда приложение завершается, то движок удаляет все шаблоны, владельцем которых было приложение.

Исключением из этого правила являются команды с префиксом **#**, если шаблон был загружен командой с таким префиксом, то он остается «жить» даже после завершения приложения-владельца.

Настройки программы

Настройки движка хранятся в файле **vgcast.cfx**. Это файл в формате XML, и его теги можно исправлять с помощью любого редактора XML-файлов (или Notepad-ом).

Журнал событий

Программа **vgcast** ведет журнал событий. Папка и префикс файла для записи журнала задается в файле конфигурации программы **vgcast.cfx**. Если содержимое тега **<LogFile>** не пустое, то журнал записывается в указанную папку, например (это значение по умолчанию, журнал включен)

```
<LogFile>. \log\vgcast</LogFile>
```

Указывает, что журнал нужно записывать в папку **log**, расположенную в той же папке, из которой запускается исполняемый файл **vgcast.exe**.

```
<LogFile>c:\temp\vgcast</LogFile>
```

Журнал будет создаваться в папке temp на диске C:. Имена файлов журнала имеют следующий вид

```
vgcast-YYMMDD.txt
```

Файл имеет текстовый формат.

Протокол обмена и кодировка текстовых данных

Обмен данными между приложениями и движком выполняется по протоколу TCP/IP. Порт обмена указан в файле vgcst.cfx. Команды движка – это текстовые строки.

Каждая строка должна завершаться байтом со значением 0.

Все текстовые данные в системе хранятся в кодировке Unicode, что позволяет работать в любой языковой среде и использовать одновременно алфавиты нескольких языков.

При передаче данных между клиентом и сервером используется кодировка UTF-8.

Именованние объектов

Все объекты движка – шаблоны и слои – имеют имена. Разделителями имен является точка. При использовании имен в командах можно употреблять символы-шаблоны (wildcards) «*» и «?». В этом случае команда применяется ко всем объектам, имена которых совпадают с учетом символов-шаблонов.

Команды управления

Если команда имеет префикс #, то она интерпретируется как глобальная команда, т.е. все объекты, которые были созданы вследствие выполнения команды, не удаляются, когда закрывается сеанс связи с движком. Это в первую очередь касается команд read и load.

exit

Завершить работу исполняющей системы

shot

Сделать слепок фрейм-буфера в файл в формате *.png. Папка для снимков задается в файле конфигурации программы vgcst.cfx. Если содержимое тега <ShotRoot> не пустое, то снимки записываются в указанную папку, например

```
<ShotRoot>.\shot\</ShotRoot>
```

Указывает, что снимки нужно записывать в папку shot, расположенную в той-же папке, из которой запускается исполняемый файл vgcst.exe.

```
<ShotRoot>c:\temp\</ShotRoot>
```

Снимки будут создаваться в папке temp на диске C:. Имена файлов-снимков имеют такой вид

```
shot-YYMMDD-HHMMSS.png
```

Формат пикселей в файле – RGB32, т.е. хранится три цветовых компоненты и альфа-канал.

replay

PRO

Эта команда позволяет включить/выключить запись всей видеоинформации, которую рендерит движок в файл в формате MOV с кодеком PNG, кодировка RGB+ALPHA.

Например, команда

```
replay c:\temp\demo.mov
```

Начинает запись в файл, а команда

```
replay
```

Завершает запись в файл.

trace level

Эта команда устанавливает режим вывода сообщений в консольное окно движка. Параметр level может принимать значения от 0 (минимальный вывод сообщений) до 2

(максимальный). Эту команду на уровне 2 удобно использовать для отладки эффектов и связи с приложениями.

Команды работы с шаблонами

read file_name

Прочитать файл с описанием шаблона. Если для слоев программы не был указан оператор **show**, то после чтения файла все его слои остаются невидимыми (полностью прозрачными) до выполнения команд **fadein** или **show**.

load file_name

Прочитать файл с описанием шаблона. Отличие от **read** состоит в том, что все слои шаблона загружаются невидимыми, вне зависимости от описания в шаблоне.

del template_name

del *

Удалить шаблон. Все слои шаблона уничтожаются и убираются из системы отображения. Если указан параметр *, то удаляются все шаблоны.

efx tpl_name.efx_name param1 param2 ... param9

Выполнить эффект **efx_name** из шаблона **tpl_name**. Параметры – это произвольные строки, которые передаются в эффект при его интерпретации. Ссылка на параметры в описании эффекта задается последовательностью \1, \2, ..., \9.

swap layer_name_1 layer_name_2

Поменять имена слоев между собой.

Клонирование шаблонов

Иногда необходимо использовать в одной сцене несколько одинаковых шаблонов, которые отличаются только своим положением на экране. В этих случаях можно описать только один шаблон и «клонировать» его. Шаблон, который будет выступать образцом для клонирования (мастер-клон) должен иметь специальное имя, в состав которого входит знак «#». Команда клонирования имеет такой синтаксис:

```
clone name[#] num x-pos y-pos
```

где **num** – это номер клона, а **x-pos** и **y-pos** – координаты клона. Например, пусть описан шаблон с такой структурой



Тогда команды

```
clone dog[#] 1 100 200
clone dog[#] 2 100 300
```

создадут два шаблона – **dog[1]** и **dog[2]** с координатами (100, 200) и (100, 300), соответственно, и к этим шаблонам можно применять любые команды, например

```
subst dog[1].breed колли
subst dog[1].name Тузик
fadein dog[1].*
```

При клонировании шаблонов можно сразу задавать значения слоев в виде

```
layer_name=value
layer_name="value"
```

Кавычки нужно использовать, если значение слоя содержит пробелы, например

```
clone dog[#] 1 100 200 breed="Ирландский сеттер" name=Том
clone dog[#] 2 100 300 breed=Лабрадор name=Митч
```

Управление прозрачностью объектов

fadein layer_name [speed]

fi layer_name [speed]

fadeout layer_name [speed]

fo layer_name [speed]

Ввести (убрать) слой микшером. Параметр speed задает количество кадров, за которое выполняется микшер. Если значение speed не указано, то эффект выполняется за один кадр (CUT).

trsp layer_name value

Задать для слоя прозрачность value от 0 (полностью прозрачный) до 256 (непрозрачный). Значение прозрачности, заданное в описании шаблона заменяется на value.

Если микшируется слой типа «видео со звуком», то при выполнении команд fi/fo изменяется и громкость соответствующей звуковой дорожки.

wipe layer_name [**wipe=**WWW [**soft=**SSS [**file=**]file_name

Задает параметры шторки.

Подстановки

subst layer_name param

Выполнить подстановку в слое с именем layer_name. Значение param зависит от типа слоя.

fsubst layer_name file_name

Выполнить подстановку в слое с именем layer_name. Содержимое подстановки выбирается из файла с заданным именем.

queue layer_name param

Поставить в очередь roll/crawl объект в слое с именем layer_name. Значение param зависит от типа слоя. Когда текущий элемент проходит область вывода (по вертикали или по горизонтали, в зависимости от типа барабана), из очереди выбирается новый объект и начинает движение по области вывода. Когда объект полностью покидает область вывода, он удаляется и все ресурсы, связанные с ним освобождаются. В очередь можно ставить объекты к слоям типа **still**, **movie**, **mpeg**, **image**, **text**, (**clock?**).

При выполнении команды **queue** инициатору команды выдаются дополнительные сообщения (описание приведено в предположении, что в очередь ставятся бегущие строки, направление справа-налево):

!qi ID – (queue init) объект поставлен в очередь и ему присвоен идентификатор ID.

!qs ID – (queue start) объект, находящийся в очереди начал отображаться, его левый край появился в области вывода.

!qn ID – (queue next) правый край отображаемого объекта появился в области вывода. Фактически это событие означает, что из очереди будет выбран следующий объект.

!qe ID – (queue end) правый край отображаемого объекта покинул область вывода.

Где ID – это уникальный идентификатор объекта, беззнаковое целое число.

push layer_name param

Поставить в очередь roll/crawl объект в слое с именем layer_name. Значение param зависит от типа слоя. Когда текущий элемент проходит область вывода (по вертикали или по горизонтали, в зависимости от типа барабана), из очереди выбирается новый объект и начинает движение по области вывода. Если в момент, когда текущий объект выведен, в очереди нет новых элементов, то барабан останавливается.

fqueue layer_name file_name

Выполнить команду queue с выборкой содержимого подстановки из файла с именем file_name.

fpush layer_name file_name

Выполнить команду push с выборкой содержимого подстановки из файла с именем file_name.

qmode layer_name (**play** | **pause**)

Останавливает или продолжает прокрутку барабана.

Расположение слоев на экране

hslide layer_name speed from_x to_x

vslide layer_name speed from_y to_y

Выполнить эффект слайд над объектом слоя. Движение объекта выполняется со скоростью speed (количество пикселей за один кадр). Используется система координат относительно левого верхнего угла прямоугольника слоя, причем значение 0 обозначает положение объекта внутри прямоугольника, +1 – объект находится справа от прямоугольника, -1 – слева. Например,

```
hslide foo.bar 2 1 0
```

объект «приедет» по горизонтали справа в прямоугольник и останется в нем.

```
hslide foo.bar 2 -1 1
```

объект «приедет» по горизонтали слева в прямоугольник и «уедет» вправо от прямоугольника.

speed ::= n | **speed**=n | **dur**=n

Скорость движения задается в пикселях за один полукадр. Длительность задается в кадрах.

hmove layer_name speed from_x to_x

vmove layer_name speed from_y to_y

Выполнить эффект слайд над объектом слоя. Движение объекта выполняется со скоростью speed (количество пикселей за один кадр). Используется система координат относительно левого верхнего угла прямоугольника слоя, указываются относительные координаты начального и конечного положения объекта. Например,

```
hmove foo.bar 2 150 50
```

объект «проедет» по горизонтали справа от координаты 150 до координаты 50.

roll layer_name speed [rep]

Выполнить вертикальную прокрутку слоя (барабан). Движение объекта выполняется со скоростью speed (количество пикселей за один полукадр). Если значение speed положительное, выполняется прокрутка снизу вверх, если отрицательное – сверху вниз. Можно задать количество повторений барабана (rep).

crawl layer_name speed [rep]

Выполнить горизонтальную прокрутку слоя (бегущая строка). Движение объекта выполняется со скоростью speed (количество пикселей за один кадр). Если значение speed положительное, выполняется прокрутка слева направо, если отрицательное – справа налево. Можно задать количество повторений бегущей строки (rep).

path layer_name speed x0 y0 x1 y1 ... xN yN

Задает кривую, по которой будет перемещаться слой. Пар x, y должно быть столько же, сколько кадров в параметре speed.

key layer_name key1 key2 ... keyN

Задает последовательность ключевых кадров. Каждый ключевой кадр может содержать такую информацию: время, координаты объекта, размеры и прозрачность. Интерполяция параметров между ключевыми кадрами может выполняться либо линейно, либо по B-сплайну, в зависимости от типа ключевого кадра. Описание ключей состоит из параметров, разделенных пробелами.

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

fNN -- ключевой кадр, выполняется сплайновая интерполяция параметров. NN задает номер кадра, для которого указаны параметры. Если номер кадра задан в виде **n+NN**, то NN задает смещение относительно *предыдущего* ключевого файла.

xNN – координата X

yNN – координата Y

hNN – ширина слоя (зум по горизонтали), нормальная ширина = 1000

vNN – высота слоя (зум по вертикали), нормальная высота = 1000. Если указано значение параметра большее 1000, то объект увеличивается, если меньше – уменьшается. Отрицательное значение приводит к «переворачиванию» объекта по вертикали.

tNN – прозрачность слоя

wNN – положение шторки (если шторка была задана в описании слоя). Значение NN может меняться от 0 до 255.

sNN – размытость края шторки (если шторка была задана в описании слоя). Значение NN может меняться от 0 до 255.

Расположение шаблонов на экране

tkey template_name key1 key2 ... keyN

PRO

Задаёт последовательность ключевых кадров, в соответствии с которыми изменяется положение шаблона на экране. Каждый ключевой кадр может содержать такую информацию: время и координаты объекта. Интерполяция параметров между ключевыми кадрами может выполняться либо линейно, либо по B-сплайну, в зависимости от типа ключевого кадра. Описание ключей состоит из параметров, разделённых пробелами.

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

fNN -- ключевой кадр, выполняется сплайновая интерполяция параметров. NN задает номер кадра, для которого указаны параметры. Если номер кадра задан в виде **n+NN**, то NN задает смещение относительно *предыдущего* ключевого файла.

xNN – координата X

yNN – координата Y

tmove template_name param=val xXs yYs xXe yYe

PRO

Задаёт перемещение шаблона из точки Xs, Ys в точку Xe, Ye. Можно задавать такие параметры:

dur=NN

Общая длительность эффекта в кадрах. По умолчанию 25 кадров.

wave=N

Количество промежуточных точек интерполяции кривой. По умолчанию равно 0, т.е. движение происходит по прямой.

left=N.N

right=N.N

Коэффициент кривизны и направление выпуклости. Если задан параметр left, то кривая выпукла влево от направления движения, если right, то вправо. Значение параметра задает расстояние промежуточной точки интерполяции по нормали к вектору из начальной к конечной точке. В качестве исходной длины нормали используется длина вектора, которая умножается на заданный коэффициент. Т.е. если значение left=1.0, то точка к которой «подтягивается» кривая будет расположена достаточно далеко, и кривизна траектории будет довольно большой.

lefta=N.N

righta=N.N

Имеет смысл использовать если количество точек интерполяции более 1. Тогда направление кривизны кривой будет последовательно меняться, например, сначала влево, затем вправо и т.д. Таким образом, можно получить волнообразную траекторию.

conv=N.N

Коэффициент затухания волн в траектории. Если значение параметра < 1, то волны будут затухать, если > 1, то будут увеличиваться.

tpos template_name XX YY

Задаёт новые координаты шаблона, на самом деле эквивалент команды

PRO

tkey template_name n1 xXX yYY

torder template_name param=val N1 N2 ... Nn

Задаёт перестановку клонов в массиве клонов

PRO

Команды слоев

type layer_name dur string

Вывести значение текстового поля эффектом «пишущей машинки». Т.е. литеры из строки string будут появляться одна за другой, с заданной задержкой (в кадрах).

PRO

text layer_name dur **key** value [format]

Довольно специфичная команда, позволяет реализовать часто используемый прием вывода информации в текстовые поля. С каждым текстовым полем связан числовой параметр. По этой команде выполняется интерполяция значения этого параметра от текущего к заданному, и непрерывный вывод этих значений в текстовое поле. Если поле формата не задано, используется формат %.0f.

clock layer_name (**play**|**stop**|**pause**|**resume**|**countdown** msec|**set** msec)

Управление часами и таймерами.

video layer_name (**play**|**pause**|**restart**|**out**)

Управление анимацией типа movie.

chart layer_name (**reset**|**render**|**add** num_point x1 y1 ...)

Управление диаграммами.

Работа со звуком

sound file_name | **sound** track_name file_name

Сыграть звуковой файл. Если не задан номер дорожки (track_name) то звук воспроизводится на дорожке 0. Звуки, воспроизводимые на разных дорожках, микшируются.

sound | **sound** track_name

Прекратить воспроизведение звука на заданной дорожке (или на умалчиваемой дорожке). Соответствующая звуковая дорожка освобождается.

si track_name dur

so track_name dur

Плавно микшировать звук за dur кадров.

Работа с объектами 3D

mode name=val name=val ...

Управление режимами работы интерпретатора

PRO

mode 3d=1 устанавливает режим преобразования всей сформированной сцены в трехмерном пространстве. Задается перспективная проекция.

dur=NN задает количество кадров, за которое будет выполняться интерполяция остальных параметров. Если значение dur не задано, то преобразования выполняются за 1 кадр.

xrot, yrot, zrot, xoff, yoff, zoff – задают повороты (в градусах) и сдвиги (в ???) по соответствующим осям. Поскольку используется перспективная проекция, то значение zoff должно быть отрицательным, причем значение около -12 соответствует примерно масштабу 1:1. Чем меньше значение zoff, тем меньший масштаб картинки. Плоскость отсечения проходит по значению оси Z = -5, поэтому объекты у которых смещение zoff больше -5 не выводятся.

Значения этих параметров могут задаваться как значения с плавающей точкой, но в этом случае нужно все значение заключать в двойные кавычки, например `zoff="-12.3"`

pie3d `layer_name dur pie_command`

PRO

Команда управления трехмерными секторными диаграммами. Параметр `dur` задает длительность эффекта в кадрах. Параметр `pie_command` задает либо поведение всего объекта, либо отдельных секторов. Интерполяция параметров выполняется за заданное количество кадров от текущего значения к указанному (линейно). Значения параметров (кроме углов) должно быть недалеко от 1.

zrot `angle` – повернуть диаграмму на угол `angle` вокруг оси Z. Угол задается в градусах.

xrot `angle` – повернуть диаграмму на угол `angle` вокруг оси X. Угол задается в градусах.

yrot `angle` – повернуть диаграмму на угол `angle` вокруг оси Y. Угол задается в градусах.

zoff `dist` – передвинуть диаграмму на заданное расстояние по оси Z.

pNN `sector_command` – команда конкретного сектора в диаграмме. NN задает номер сектора от 0 до максимального количества секторов в диаграмме. Если NN не указано, или задана команда `p*`, то команда применяется ко всем секторам диаграммы.

p0 ro NN – изменить внешний радиус сектора 0 до величины NN.

p0 ri NN – изменить внутренний радиус сектора 0 до величины NN (радиус дырки).

p0 ht NN – изменить высоту сектора 0 до величины NN.

p0 ex NN – изменить отступ сектора 0 от центра до величины NN.

p0 wa NN – изменить угол сектора 0 до величины NN.

p0 sa NN – изменить начальный угол сектора 0 до величины NN.

p0 ct 0xAARRGGBB – изменить цвет сектора 0 до указанного значения.

bar3d `layer_name dur bar_command`

PRO

Команда управления трехмерными столбчатыми диаграммами. Параметр `dur` задает длительность эффекта в кадрах. Параметр `bar_command` задает либо поведение всего объекта, либо отдельных столбиков. Интерполяция параметров выполняется за заданное количество кадров от текущего значения к указанному (линейно).

p0 ro NN – изменить размер основания столбика по оси X до величины NN. NN – это числа с плавающей точкой, вообще говоря их значения должны быть недалеко от 1.0

p0 ht NN – изменить размер основания столбика по оси Y до величины NN.

p0 ri NN – изменить смещение столбика по оси X до величины NN. Может быть отрицательной или положительной величиной

p0 ex NN – изменить смещение метки столбика по оси Y до величины NN.

p0 sa NN – изменить смещение основания столбика по оси Y до величины NN.

p0 wa NN – изменить высоту столбика по оси Y до величины NN.

p0 ct 0xAARRGGBB – изменить цвет верхушки столбика.

p0 cb 0xAARRGGBB – изменить цвет основания столбика.

Информационные команды

help [`command_name`]

Выдать короткую справку по командам

sts display

Выдать состояние системы отображения

sts layer

Выдать состояние всех слоев шаблонов

sts comm

Выдать состояние коммуникационной подсистемы

sts template

Выдать состояние шаблонов

Для информационных команд желательно указывать точное имя слоя, без *. Если указано расширенное имя слоя, то на один запрос будет выдано несколько строк с информацией о слоях, но определить к какому слою относится какая информация невозможно.

info pos layer_name

Выдать текущие координаты слоя. Ответ передается в виде текстовой строки

```
pos XXX YYY
```

где XXX YYY – координаты левого верхнего угла объекта относительно области вывода слоя. Могут быть как положительные значения, так и отрицательные.

info size layer_name

Выдать текущие размеры слоя. Ответ передается в виде текстовой строки

```
size WWW HHH
```

где WWW HHH – текущие размеры объекта. Команда полезна для объектов, размер которых вычисляется во время выполнения, например для текстовых объектов.

info trsp layer_name

Выдать текущую прозрачность слоя. Ответ передается в виде текстовой строки

```
trsp TTT
```

где TTT – текущая прозрачность объекта. 0 = объект полностью прозрачный (невидимый), 256 = объект полностью непрозрачный.

info rect layer_name

Выдать область вывода слоя. Ответ передается в виде текстовой строки

```
rect BX BY EX EY
```

где BX – абсолютная координата левого угла области вывода, BY – абсолютная координата верхнего угла области вывода, EX – абсолютная координата правого края области вывода+1, EY – абсолютная координата нижнего края области вывода+1. Таким образом, ширина области вывода $WD = EX - BX$, высота области вывода $HT = EY - BY$.

Описание шаблона

Поскольку формирование шаблонов выполняется с помощью программы VgEdit, описание шаблонов приведено скорее для внутреннего использования.

Каждая команда описания шаблона должна располагаться на отдельной строке. Комментарий начинается со знаков // и продолжается до конца строки. Имена шаблонов и слоев могут состоять из произвольных знаков, включая пробелы.

Шаблон состоит из произвольного количества слоев. Порядок описания слоев задает их порядок на экране (z-order), самый первый слой является самым нижним, самый последний слой – самый верхний. В каждом слое задается его геометрия (размеры и положение на экране) и описывается тип слоя – заливка сплошным цветом, изображение, анимация, часы и т.д. Никаких ограничений на взаимное расположение слоев нет, они могут перекрываться.

Можно явно задавать приоритет слоя. Это принципиально важно для тех слоев, которые априори должны располагаться поверх других – часы, логотипы и т.д.

Каждый шаблон имеет имя, которое используется для ссылки на элементы шаблона. Имя шаблона указывается командой

tnam template_name

template template_name

где `template_name` произвольная последовательность литер (включая пробелы) до конца строки или до комментария. Если команда `tnam` отсутствует в шаблоне, то шаблон имеет такое же имя, как и файл шаблона.

Каждый слой описывается группой команд, начало слоя задается командой `layer` (**layer**), завершает описание слоя команда `elay` (**endlayer**).

layer

команда описания слоя

...

команда описания слоя

endlayer

Некоторые команды описания слоя являются общими для всех типов слоев, некоторые специфичны для конкретных типов слоев.

Общие операторы описания слоя.

name layer_name

Команда задает имя слоя, это произвольная последовательность литер (включая пробелы) до конца строки или до комментария. Вообще говоря имя слоя может отсутствовать, его необходимо задавать только если нужно менять параметры слоя в процессе работы (например, менять его прозрачность или положение на экране) или выполнять подстановки.

type layer_type

Команда задает тип слоя, где `layer_type` может принимать следующие значения

soli | **solid**

– слой заполняется сплошным цветом

grad | **gradient**

– слой заполняется градиентной заливкой

imag | **image**

– слой содержит изображение. Допустимые форматы картинок TGA, BMP, VII, VIS, VIM.

still

– слой содержит изображение, все операции с слоем такого типа совпадают с форматом **image**. Допустимые форматы картинок – все те, которые поддерживаются системой DirectX Microsoft Windows, например TGA, BMP, JPG, GIF, AVI, и т.д. Если указаны форматы файлов, содержащие видеoinформацию, то в качестве картинки используется первый кадр видеопотока. Для того, чтобы можно было читать и декодировать изображения, в DirectX должен быть установлен соответствующий кодек. Так, например, для чтения файлов в формате MPEG-2 нужно установить какой-либо из программных декодеров MPEG-2, например Elecard.

movi | **movie**

– слой содержит анимацию с альфа-каналом

mpeg

– слой содержит анимацию в формате MPEG-1/2

text

– слой содержит форматированный текст

clck | **clock**

– слой содержит часы или таймер

chart

– слой содержит диаграмму

В зависимости от типа слоя в описании слоя могут присутствовать дополнительные команды.

rect left top width height

Эта команда описывает геометрию слоя – его позицию на экране и размер области. Размер слоя может не совпадать с размером объекта, формирующего слой.

trsp number

transparency number

Прозрачность слоя, может изменяться от 0 до 256. Если эта команда не задана, то по умолчанию прозрачность слоя устанавливается в 256 – полностью непрозрачный слой. Если прозрачность слоя задана, то команды **fade** не могут установить ее больше чем **number**.

pri number | **min** | **max**

priority number | **min** | **max**

Приоритет слоя, может принимать любое целочисленное значение, как положительное, так и отрицательное. Значение **min** соответствует минимальному приоритету (самый нижний слой), **max** – максимальному приоритету (самый верхний слой). Если эта команда не задана, то по умолчанию приоритет слоя устанавливается в 0. Слои с одинаковыми приоритетами располагаются в порядке загрузки слоев, самый последний загруженный слой становится самым верхним.

pos x-coord y-coord

Исходные координаты объекта, формирующего слой относительно прямоугольника слоя. Для использования команд **slide** или **roll/crawl**, чтобы исходное положение объекта было за пределами прямоугольника слоя.

show

Исходное состояние прозрачности слоя. По умолчанию все слои создаются полностью прозрачными, и становятся доступны для отображения только после команд **show** или **fadein**. Если же в описании слоя указан оператор **show**, то слой становится видимым сразу после создания (чтения файла с шаблоном).

Операторы, специфичные для конкретных слоев

file file_name

Команда **file** задает имя файла, содержащего объект описывающий слой. Для слоя типа **image** это графический файл в формате TGA, BMP, VII, VIS или VIC. Для слоя типа **movie** – файл содержащий анимацию в формате VIM. Для слоя типа **clock** – описание часов в формате CLK.

alpha number

Команда **alpha** принудительно задает прозрачность (альфа-канал) для изображений типа **still** (DirectX).

fit (0 | 1)

Команда **fit 1** указывает, что изображение нужно масштабировать до размеров области вывода слоя (в текущей реализации – только в сторону уменьшения).

hpos (0 | 1 | 2)

Команда **hpos** указывает, как изображение выравнивается относительно области вывода по горизонтали. Значение 0 указывает, что выравнивание происходит по левому краю области вывода, значение 1 – по правому краю, значение 2 – по центру области. Применимо для слоев типа **image** и **still**.

vpos (0 | 1 | 2)

Команда **vpos** указывает, как изображение выравнивается относительно области вывода по вертикали. Значение 0 указывает, что выравнивание происходит по верхнему краю области вывода, значение 1 – по нижнему краю, значение 2 – по центру области. Применимо для слоев типа **image** и **still**.

vflip

Команда **vflip** переворачивает изображение по вертикали. Связано с тем, что некоторые файлы в формате TGA используют загадочную привязку левого верхнего угла картинки.

orig template.layer

Команда **orig** указывает, что координаты прямоугольника текущего слоя заданы относительно левого верхнего угла прямоугольника того слоя, на который ссылается команда. Это оператор должен быть указан после команды **rect**.

fldo field_order

field field_order

Задаёт порядок полей для анимации. Значение 0 указывает, что в файле анимации первым идет четное поле, затем нечетное (синонимы -- Upper field first = Top field first = even = field order B). Значение 1 указывает обратный порядок полей -- Lower field first = Bottom field first = odd = field order A.

loop number | max

Количество повторений анимации. Значение 0 или -1 задает бесконечное количество повторений.

colr (number | **rgb**=number | **yuv**=number)

color (number | **rgb**=number | **yuv**=number)

Цвет слоя для слоев типа **solid**. Если число начинается со знаков **0xNNNNNNNNN** то цвет задан шестнадцатеричным числом, в противном случае -- десятичным.

play

Начальное состояние слоя часов – запущены (так, как будто была выполнена команда **clock * play**). Вообще говоря, не имеет смысла для таймеров с обратным отсчетом.

pause

Начальное состояние слоя **movie** – пауза (так, как будто была выполнена команда **movie * pause**).

Примеры описания слоев в шаблоне

```
template test // имя шаблона
```

```
layer // заливка прямоугольника сплошным цветом
```

```
name цвет // имя слоя
type solid // тип слоя
rect 50 100 200 50 // положение и размеры прямоугольника
color 0xffff0000 // цвет прямоугольника (красный)
trsp 128 // прозрачный на 50%
```

```
endlayer
```

Допустимы любые команды слоя: **fadein**, **fadeout**, **show**, **hide**, **pos**, **hslide**, **vslide**, **crawl**, **roll**. Команда **subst** позволяет изменить цвет прямоугольника. Например:

```
subst test.цвет 0xff00ff00
```

изменит цвет прямоугольника на зеленый.

```
layer // изображение
```

```
name картинка // имя слоя
type image // тип слоя
rect 50 50 400 250 // положение и размеры прямоугольника
file "c:\images\try.tga" // файл с картинкой
```

```
endlayer
```

Допустимы любые команды слоя: **fadein**, **fadeout**, **show**, **hide**, **pos**, **hslide**, **vslide**, **crawl**, **roll**. Команда **subst** позволяет изменить содержимое прямоугольника. Например:

```
subst test.картинка "c:\images\new.tga"
```

изменит содержимое прямоугольника на содержимое файла **new.tga**.

```
layer // изображение в формате DirectX
```

```
name картинка // имя слоя
type still // тип слоя
```

```

rect 50 50 400 250 // положение и размеры прямоугольника
file "c:\images\try.jpg" // файл с картинкой

```

endlayer

Изображения в формате DirectX могут содержать в себе альфа-канал (слой прозрачности). Если нужно слой прозрачности игнорировать, можно использовать оператор `alpha = NN`, где `NN` может принимать значения от 0 (полностью прозрачная картинка) до 255 (полностью непрозрачная картинка).

```

layer // анимация с альфа-каналом
name "аним лого" // имя слоя
type movie // тип слоя
rect 50 50 320 240 // положение и размеры прямоугольника
file "c:\movie\alogo.vim" // файл с картинкой
field 1 // порядок полей – нечет
loop max // повторять до бесконечности
pri max // самый верхний слой

```

endlayer

Допустимы любые команды слоя: **fadein**, **fadeout**, **show**, **hide**, **pos**, **hslide**, **vslide**, **crawl**, **roll**. Команда **subst** позволяет изменить выполняющуюся в данный момент анимацию. Например:

```
subst test."аним лого" "c:\movie\new_logo.tga"
```

изменит содержимое прямоугольника на содержимое файла `new_logo.tga`.

```

layer // файлы в формате MPEG-2
name клип1 // имя слоя
type mpeg // тип слоя
rect 0 0 720 576 // положение и размеры прямоугольника
file "c:\clips\mbg.mpg" // файл с картинкой
field 1 // порядок полей – нечет
loop max // повторять до бесконечности
pri min // самый нижний слой

```

endlayer

Допустимы любые команды слоя: **fadein**, **fadeout**, **show**, **hide**, **pos**, **hslide**, **vslide**, **crawl**, **roll**. Команда **subst** выполняется так-же, как и для слоя `movie`.

```

layer // часы (таймер и т.д.)
name clock // имя слоя
type mpeg // тип слоя
rect 50 30 100 50 // положение и размеры прямоугольника
file "c:\clocks\dig.clk" // файл с описанием часов
pri max // самый верхний слой

```

endlayer

Допустимы любые команды слоя: **fadein**, **fadeout**, **show**, **hide**, **pos**, **hslide**, **vslide**, **crawl**, **roll**. Команда **subst** выполняет замену часов (таймера) в соответствии с новым описанием часов.

```
subst test.clock "c:\clocks\analog.clk"
```

Для часов есть дополнительная команда, управляющая состоянием часов/таймеров.

```

clock test.clock play -- включить часы (таймер)
clock test.clock stop -- выключить часы (таймер)
clock test.clock pause -- остановить отсчет таймера
clock test.clock resume -- возобновить отсчет таймера
clock test.clock countdown msec -- установить значение обратного отсчета для
таймера (в миллисекундах)

```

```

layer // диаграмма
name ch
type chart
rect 100 100 500 200

```

```

color 0xff0000ff // цвет (синий)
size 2 // толщина линии (0 – заливка)
focus "dot.tga" // отмечать конечную точку кривой картинкой
shape 0 // форма 0 – ломаная, 1 – кривая, 2 - столбики

```

endlayer

Допустимы любые команды слоя: **fadein**, **fadeout**, **show**, **hide**, **pos**, **hslide**, **vslide**, **crawl**, **roll**. Команда **subst** для диаграмм неприменима. Дополнительная команда **chart** управляет состоянием вывода диаграмм:

```

chart test.ch reset -- сбросить количество точек на диаграмме в 0
chart test.ch render -- отрисовать текущее состояние диаграммы
chart test.ch add 4 0 100 40 50 80 150 120 80
-- добавить к диаграмме 4 точки, первая точка X = 0, Y = 100, вторая X = 40, Y = 50 и т.д.
Все координаты указываются относительно прямоугольника слоя. Начало системы
координат левый верхний угол прямоугольника.

```

Например, такая последовательность команд:

```

chart test.ch reset
chart test.ch add 4 0 100 40 50 80 150 120 80
chart test.ch render
chart test.ch add 1 160 120
chart test.ch render

```

сначала отрисует ломаную линию из четырех точек, затем добавит еще одну и отрисует результирующую ломаную из пяти точек.

Операторы описания текстовых слоев

Описание стилей могут располагаться в любом месте файла до использования ссылки на стиль.

```

style style_name
    команда описания стиля
    ...
    команда описания стиля

```

endstyle

paragraph style_name
Ссылка на стиль. Последующий текст приобретает все атрибуты стиля.

roll roll_speed

Текстовый слой для задания последовательности барабанов. Для этого слоя **не нужно** задавать оператор **text**. Если **roll_speed** > 0 то блок текста двигаются снизу-вверх, если **roll_speed** < 0, то сверху-вниз. Подстановка текстовой информации в этот слой выполняется специальной командой **queue**.

```
queue layer_name text
```

Текст становится в очередь к слою, и когда текущий текстовый элемент проходит область вывода слоя, из очереди выбирается очередной текстовый элемент. Таким образом можно формировать непрерывные последовательности текстовых блоков, проходящих область вывода по вертикали снизу-вверх или сверху-вниз.

Информационные команды (**info pos**, **info size** и т.д.) применительно к слою **roll** выдают информацию о последнем активном элементе.

Вообще говоря, в основном использование текста типа **roll** имеет смысл для текста с параметром **wrap**.

crawl crawl_speed

Текстовый слой для задания последовательности бегущих строк. Для этого слоя **не нужно** задавать оператор **text**. Если **crawl_speed** > 0 то бегущие строки двигаются справа-налево, если **crawl_speed** < 0, то слева направо. Подстановка текстовой информации в этот слой выполняется специальной командой **queue**.

```
queue layer_name text
```

Текст становится в очередь к слою, и когда текущий текстовый элемент проходит область вывода слоя, из очереди выбирается очередной текстовый элемент. Таким образом можно формировать непрерывные последовательности бегущих строк.

Информационные команды (`info pos`, `info size` и т.д.) применительно к слою `crawl` выдают информацию о последней активной строке.

- font** font_name
Задаёт имя шрифта.
- face** num
Задаёт тип шрифта. 0 – обычный текст (plain), 1 – жирный (bold), 2 – наклонный (italic), 3 – жирный наклонный (bold italic)
- caps** num
Способ применения заглавных литер. 0 – обычные литеры, 1 – small caps, 2 – initial caps, 3 – all caps
- size** num
Задаёт размер шрифта.
- chsoft** num
Задаёт размытый край у литер.
- chcolor** num
Задаёт цвет литер.
- shsoft** num
Задаёт размытый край у тени.
- shcolor** num
Задаёт цвет тени.
- shxoff** num
Задаёт смещение тени по X.
- shyoff** num
Задаёт смещение тени по Y.
- bdsoft** num
Задаёт размытый край у бордюра.
- bdcolor** num
Задаёт цвет бордюра.
- bdsizе** num
Задаёт толщину бордюра.
- tracking** num
Задаёт межлитерное расстояние.
- scale** num
Задаёт горизонтальный масштаб.
- align** num
Выравнивание по горизонтали. 0 – влево, 1 – вправо, 2 – центрировать, 3 – выровнять по обоим краям.
- leading** num
Межстрочное расстояние.
- base** num
Сдвиг базовой линии.
- wrap** num
Если 1 – выполнять перенос строк если текст не помещается в прямоугольник. Перенос выполняется по границе слова.

Примеры описания текстовых слоев в шаблоне

Вообще говоря, если не задано описание стиля или описание формата текста соответствующими командами, то для форматирования текста используется стиль «по умолчанию» с такими

атрибутами: шрифт = Arial, размер = 24, цвет = белый непрозрачный, выравнивание = по левому краю.

```
layer // текстовый слой
  name txt1 // имя слоя
  type text // тип слоя
  rect 0 0 400 35 // положение и размеры прямоугольника
  text Тестовая строка
endlayer // формирует текст Arial, 24, белый, выравнивание влево
layer // текстовый слой
  name txt2 // имя слоя
  type text // тип слоя
  rect 0 0 400 35 // положение и размеры прямоугольника
  text Тестовая строка
  bdsizе 2 // использовать окантовку шириной 2 пиксела
  bdsoft 2 // с «размытым» краем
  bdcolor 0xff000000 // цвет черный
endlayer
```

Пример использования описания стиля:

```
style my // определяем стиль
  font Times New Roman // шрифт
  face 1 // начертание - жирное
  size 40 // размер 40 пикселей
  chcolor 0xffffffff00 // цвет литер - желтый
  bdcolor 0xff000000 // окантовка - черный
  bdsizе 2 // размер окантовки 2 пиксела
  bdsoft 2 // «мягкий» край
endstyle
layer
  name txt
  type text
  rect 100 50 500 200
  paragraph my // использовать для форматирования стиль
  align 2 // выравнивать по центру
  text Привет, world
endlayer
```

В командах подстановки для текстовых слоев необходимо указывать новый текст:

```
subst test.txt А это новый текст вместо старого
```

Описание эффектов

Описание эффектов используется для взаимодействия с редактором шаблонов. Интерпретатор эти операторы не использует. Описания эффектов должны следовать непосредственно за оператором задания имени шаблона.

\$efx effect_name key modifier

Задаёт имя эффекта **effect_name** (если используются пробелы в имени, то в двойных кавычках). Параметр **key** задаёт десятичный код «горячей» клавиши, присвоенной эффекту, а параметр **modifier** десятичное значение модификатора CTRL, ALT или SHIFT.

\$cmd delay command

Описание команд, формирующих эффект. Параметр **delay** задаёт задержку команды в кадрах относительно начала эффекта.

Пример организации связи

```
// start telnet client
//          if name == NULL -- connect to localhost
//          if name == hostname -- try to find host by name
//          if name == N.N.N.N      -- try to find host by addr
SOCKET
tnc_start(char *name, int port) {
    WSADATA wsaData;
    struct hostent *hp;
    unsigned int addr;
    char *server_name;
    struct sockaddr_in server;
    SOCKET conn_socket;
    int socket_type = SOCK_STREAM;

    if (name == NULL)
        server_name = "localhost";
    else
        server_name = name;

    if (WSAStartup(0x202,&wsaData) == SOCKET_ERROR) {
        WSACleanup();
        return NULL;
    }

    // Attempt to detect if we should call gethostbyname() or
    // gethostbyaddr()

    if (isalpha(server_name[0])) { /* server address is a name */
        hp = gethostbyname(server_name);
    }
    else { /* Convert nnn.nnn address to a usable one */
        addr = inet_addr(server_name);
        hp = gethostbyaddr((char *)&addr,4,AF_INET);
    }
    if (hp == NULL ) {
        WSACleanup();
        return NULL;
    }

    // Copy the resolved information into the sockaddr_in structure
    memset(&server,0,sizeof(server));
    memcpy(&(server.sin_addr),hp->h_addr,hp->h_length);
    server.sin_family = hp->h_addrtype;
    server.sin_port = htons(port);

    conn_socket = socket(AF_INET,socket_type,IPPROTO_TCP); // Open a socket
    if (conn_socket < 0) {
        WSACleanup();
        return NULL;
    }

    //
    // Notice that nothing in this code is specific to whether we
    // are using UDP or TCP.
    // We achieve this by using a simple trick.
    // When connect() is called on a datagram socket, it does not
    // actually establish the connection as a stream (TCP) socket
    // would. Instead, TCP/IP establishes the remote half of the
    // ( LocalIPAddress, LocalPort, RemoteIP, RemotePort) mapping.
    // This enables us to use send() and recv() on datagram sockets,
    // instead of recvfrom() and sendto()
```

```

        if (connect(conn_socket, (struct sockaddr*)&server, sizeof(server))
            == SOCKET_ERROR) {
            WSACleanup();
            return NULL;
        }

        return conn_socket;
    }

// stop telnet client
void
tnc_stop(SOCKET socket) {
    closesocket(socket);
    WSACleanup();
}

// send data from client to server
int
tnc_send(SOCKET socket, char* buf) {
    int retval;
    char msgbuf[TELNET_MAXMSG];
    char* bp = buf;
    char prev = 0;
    int msglen = 0;

    if (tnc == NULL) return 0;
    if (bp == NULL) return 0;

    // translate buf into msgbuf
    while (*bp != 0) {
        if (*bp == '\n' && prev != '\r') {
            msgbuf[msglen++] = '\r';
        }
        msgbuf[msglen++] = prev = *bp++;
    }
    if (prev != '\n') {
        msgbuf[msglen++] = '\r';
        msgbuf[msglen++] = '\n';
    }
    msgbuf[msglen++] = 0;

    retval = send(socket, msgbuf, msglen, 0);
    if (retval == SOCKET_ERROR)
        return -1;

    return retval;
}

```