|  |
| --- |
| SG Bitmap Font Editor |
| Использование скриптов для экспорта шрифтов |
| Если возможностей редактора как графического приложения недостаточно для экспорта шрифта в желаемом формате, шрифт может быть выгружен в виде данных путём написания скрипта. Последний получит данные путём обращения к редактору как к COM серверу. Этот документ о том, как это сделать. |

*(с) Григорьев Вячеслав Владимирович*

2004-2016

[[1]](#footnote-1)#Использование скриптов для экспорта шрифтов

Как может показаться в первый момент, возможности редактора по экспорту шрифтов сильно ограничены. Такое мнение может сложиться из-за просмотра окошка экспорта, имеющего сравнительно немного элементов управления собственно экспортом. Однако помимо этого окошка редактор предоставляет ещё одну, гораздо более мощную возможность по экспорту. А именно – написание и использование скриптов экспорта.

Поддержка скриптов является неотъемлемой частью операционной системы Windows, начиная с системы, имеющей Internet Explorer 3.0 и выше. По умолчанию в системе имеется поддержка Java Script’а (JScript) и Visual Basic Script’а (VBScript). Т.е., Вы можете описать требуемые действия в текстовом файле с расширением, соответственно, \*.js или \*.vbs (по умолчанию) и запустить его на исполнение.

В скриптах имеется возможность создания и управления COM-объектами, зарегистрированными в системе. Именно она и открывает широкие возможности по использованию скриптов. Работа с объектами ведётся с помощью их свойств и методов. Редактор предоставляет несколько зарегистрированных COM-объектов, с помощью которых можно управлять вариантами хранения графических данных открытого в настоящий момент шрифта, и по желанию сохранять их в файл экспорта. **Примечание:** для использования COM-объектов редактора они должны быть зарегистрированы в системе. Инсталлятор редактора делает это автоматически. Однако, если Вы им не пользовались, то для успешной работы со скриптами необходимо выполнить команду «sgfed.exe /RegServer» в командной строке. Перед удалением редактора совсем не лишним будет выполнить команду «sgfed.exe /UnregServer» для очистки реестра.

Замечу, что работа с редактором через скрипты гораздо легче, чем может показаться на первый взгляд. Это подтвердят примеры скриптов, приведённые после описания объектной модели. Все дальнейшие примеры будут приведены (да простят меня пользователи) только для JScript’а.

Объектная модель сервера редактора

Главный объект, предоставляемый редактором, называется “SGFED.SGBitmapFontServer”. Он предоставляет доступ к самому редактору и открытому в нём в настоящий момент шрифту. Любая работа в скрипте для экспорта шрифтов так или иначе начинается с создания сущности этого объекта (или, если не вдаваться в терминологию ООП, просто «создания объекта»).

var Srv = WScript.CreateObject("SGFED.SGBitmapFontServer");

Где WScript – глобальный объект в JScript’е, предоставляющий сервис коду скрипта к основным базовым операциям. Он не нуждается в какой-либо дополнительной инициализации.

Для начала работы с конкретным символом с помощью этого объекта необходимо создать объект символа:

var MySymbol = Srv.GetSymbolObject(10);

В этом примере мы создали объект, содержащий графические данные символа номер 10 в текущем открытом шрифте в редакторе. Данные, пока мы не произведём дальнейших действий, хранятся в нём в формате, описанном здесьSGBitmapFontServer\_GetSymbolObject.

Дальнейшие действия с данными осуществляются с помощью методов объекта SGBitmapSymbol, сущность которого мы только что создали. Их полный список можно посмотреть в оглавлении. Пусть, для примера, у нас имелся символ с размером 8 \* 10 пикселей. Это значит, что в памяти он по ширине занимает два байта, причём во втором биты выровнены к левому краю. Допустим, нам надо к правому. Воспользуемся следующей функцией:

MySymbol.CopyRect(8, 0, 2, 8, 14, 0);

Что означает, что мы скопировали область с координатами left = 8, top = 0, width = 2, height = 8 в область с координатами левого верхнего края left = 14, top = 0. Следует заметить, что везде, где речь идёт о битовых областях, по умолчанию считается, что указывается левый верхний край области и её ширина и высота. Если нам не нужна «грязь» там, откуда мы только что скопировали биты, очистим её:

MySymbol.ClearRect(8, 0, 2, 8);

Дальнейшие действия по преобразованию данных перед экспортом могут включать в себя различные повороты, отражения, смещения данных. А можно сделать всё, что необходимо «ручками», воспользовавшись «сырыми» данными посредством свойств DataBytes, DataWords или DataPixels объекта SGBitmapSymbol.

Для записи подготовленных данных можно воспользоваться предлагаемыми тем или иным скриптовым языком механизмами. К примеру, ниже приведён простейший скрипт для экспорта шрифта размером символов 8\*8 пикселей в C-файл. К слову говоря, на моей машине с процессором Pentium 4 1600 MHz он выполняется около 7 секунд в основном благодаря неимоверной скорости интерпретации текста.

var Srv = new ActiveXObject("SGFED.SGBitmapFontServer");

var Symbol = Srv.GetSymbolObject(0);

var fso = new ActiveXObject("Scripting.FileSystemObject");

var file = fso.CreateTextFile(Srv.FontFileName + ".c", true);

// Write prologue

file.WriteLine('// This is the sample export file, generated from JScript');

file.WriteLine('// SymbolsCount = ' + Srv.SymbolsCount + ', Width = ' +

 Srv.SymbolWidth + ', Height = ' + Srv.SymbolHeight + '\n');

file.WriteLine('unsigned char Table[] = {');

// Write body

for(var Index = 0; Index < Srv.SymbolsCount; Index++)

{

 var Line = '\t';

 Symbol.LoadSymbol(Index);

 for(var ByteIndex = 0; ByteIndex < Symbol.Height; ByteIndex++)

 {

 if(ByteIndex != 0) Line += ', ';

 Line += '0x';

 if(Symbol.DataBytes(ByteIndex).toString(16).length == 1) Line += '0';

 Line += Symbol.DataBytes(ByteIndex).toString(16);

 }

 if(Index < Srv.SymbolsCount - 1) Line += ', ';

 file.WriteLine(Line);

}

// Write epilogue

file.WriteLine("};");

file.Close();

Srv.ShowMessage('Export done.');

Для проверки этого скрипта скопируйте его в текстовый файл с расширением \*.js и запустите его с командной строки. Естественно, при этом должен быть открыт редактор, и в него должен быть загружен шрифт.

Ещё один пример скрипта делает тоже самое, что и экспорт с помощью опции «Bits Scanning: Vertical (String scanning)» в окошке экспорта редактора:

var Srv = new ActiveXObject("SGFED.SGBitmapFontServer");

var Symbol = Srv.GetSymbolObject(0);

var fso = new ActiveXObject("Scripting.FileSystemObject");

var file = fso.CreateTextFile("BigFont.as", true);

// Write prologue

file.WriteLine(';Warning! This file generated automatically by SGFED\n');

file.WriteLine('\tpsect\ttext, class=CODE, reloc=2');

file.WriteLine('\tGLOBAL\t\_dspBigFont\n');

file.WriteLine('\_dspBigFont:');

// Write body

var Line;

for(var Index = 0; Index < Srv.SymbolsCount; Index++)

{

 Symbol.LoadSymbol(Index);

 Symbol.MirrorRect(0, 0, 12, 16, true);

 Symbol.Rotate();

 Symbol.RemoveLines(12, 4);

 Symbol.Rotate();

 Symbol.Rotate();

 for(var ByteIndex = 0; ByteIndex < Srv.SymbolWidth; ByteIndex++)

 {

 if(ByteIndex != 0) Line += ', ';

 else Line = '\tdb ';

 Line += Symbol.DataBytesStr(ByteIndex \* 2, 1);

 }

 file.WriteLine(Line);

 for(var ByteIndex = 0; ByteIndex < Srv.SymbolWidth; ByteIndex++)

 {

 if(ByteIndex != 0) Line += ', ';

 else Line = '\tdb ';

 Line += Symbol.DataBytesStr(ByteIndex \* 2 + 1, 1);

 }

 file.WriteLine(Line);

}

file.Close();

Srv.ShowMessage('Export done.');

Это мой рабочий пример для шрифта размером 12\*16 пикселей. Естественно, что он выполняется не мгновенно, как, к примеру, функция экспорта в редакторе, а около 10 секунд. И это несмотря на то, что здесь применены строковые свойства DataBytesStr объекта SGBitmapSymbol, чтобы избежать конвертирования в самом скрипте, как в примере выше.

Вопросы быстродействия

При выполнении скриптов работают следующие механизмы: интерпретатор скриптов, RPC-вызов процедур и, собственно, сам редактор. Как показала практика, наиболее «тормозным» является интерпретатор скриптов. Единственное разумное, что здесь можно предложить – постараться как можно более компактно писать скрипт и как можно меньше использовать циклы. Те, кто знаком с каким-либо языком, позволяющим использовать COM-объекты (в более общем случае – поддерживающим OLE-автоматизацию), могут писать вызовы редактора на этом языке. Это будет несомненно быстрее. Как вариант – C++, Delphi, C#, Visual Basic и другие.

В заключении отмечу, что в будущем будут добавляться новые функции редактирования, позволяющие нормальным путём делать то, что раньше, возможно, приходилось делать скриптами. Ну а экспорт – как правило, операция редкая. Можно и подождать.

[[2]](#footnote-2)#Объект SGBitmapFontServer

### Методы:

GetSymbolObject(symbol)

Создание объекта, отвечающего за символ с номером, указанным параметром “symbol”. Объект имеет свою копию графических данных, в дальнейшем никак не связанную с символом в шрифте. Данные могут храниться по строкам или по столбцам. Вариант хранения задаётся при создании объекта символа и в последующем не может быть изменён. Однако расположение битов позже можно будет менять по своему усмотрению.

В горизонтальном варианте хранения данные содержатся построчно. Выровнены на

Данные содержатся построчно, выровнены на границу байта к левому краю. С помощью методов объекта SGBitmapSymbol их можно будет переставлять по желанию. А на данном этапе они располагаются так:



Пример вызова:

var MySymbol = Srv.GetSymbolObject(10);

[[3]](#footnote-3)#Объект SGBitmapFontServer

### Методы:

ShowMessage(msg)

Отображает во всплывающем окне сообщение, указанное переменной msg. Это может быть целочисленная, строковая переменная, или переменная с плавающей запятой. Используя возможности скрипта по части автоматического преобразования типов данное окошко можно легко использовать в целях отладки скрипта. Например, для вывода размеров символа:

Srv.ShowMessage(MySymbol.Width + ‘ \* ‘ + MySymbol.Height);

В данном примере в функцию ShowMessage() передаётся строка, содержащая размеры символа MySymbol.

[[4]](#footnote-4)#Объект SGBitmapFontServer

### Методы:

SaveUndoState()

Эта функция сохраняет текущее состояние шрифта в очереди операций отмены. Для экспорта она не нужна, но может помочь при использовании скриптов во время редактирования шрифта в редакторе.

[[5]](#footnote-5)#Объект SGBitmapFontServer

### Свойства:

FontFileName

Строковое свойство, содержащее полный путь открытого в настоящий момент документа со шрифтом.

SymbolsCount

Целочисленное свойство, указывающее количество символов в открытом в редакторе шрифте.

SymbolWidth

Ширина символов текущего шрифта в пикселях. В отличие от подобного свойства у объектов типа SGBitmapSymbol эта ширина не выравнивается на границу байта.

SymbolHeight

Высота символов текущего шрифта в пикселях.

Version

Версия реализации COM-объектов редактора. Т.к. в будущём, возможно, эти объекты будут дополняться новыми свойствами и методами, с помощью этого свойства можно контролировать, подходит ли установленная версия редактора для выполнения скрипта. Текущая версия – 2.

[[6]](#footnote-6)#Объект SGBitmapSymbol

### Методы:

CopyRect(left, top, width, height, new\_left, new\_top)

Копирует битовую область, указанную в пикселах координатами left, top, width, height в новое место, указанное координатами new\_left, new\_top. Случай перекрытия областей проверяется и правильно отрабатывается.



Пример вызова:

MySymbol.CopyRect(8, 0, 2, 8, 14, 0);

[[7]](#footnote-7)#Объект SGBitmapSymbol

### Методы:

ClearRect(left, top, width, height)

Заполняет указанную область нулями.

[[8]](#footnote-8)#Объект SGBitmapSymbol

### Методы:

Rotate()

Поворачивает плоскость на 90 градусов по часовой стрелке. Так как по горизонтали плоскость всегда занимает ровное количество байт, при необходимости перед поворотом она расширяется по высоте. Схематично преобразования можно представить так:



Таким образом, новая высота будет равна старой ширине, а новая ширина – старой высоте, округлённой в большую сторону к границе байта. Для удаления лишних строчек по высоте можно воспользоваться функцией RemoveLines(…)SGBitmapSymbol\_RemoveLines.

[[9]](#footnote-9)#Объект SGBitmapSymbol

### Методы:

RemoveLines(StartLine, LinesCount)

Эта функция удаляет горизонтальные строки в графическом образе символа в объекте. Таким образом, ширина символа остаётся прежней, а высота уменьшается на значение параметра LinesCount.

[[10]](#footnote-10)#Объект SGBitmapSymbol

### Методы:

MirrorRect(Left, Top, Width, Height, bHorizontal)

Эта функция производит отражение в заданной прямоугольной области. Если bHorizontal = true, происходит горизонтальное отражение. В противном случае – вертикальное.

[[11]](#footnote-11)#Объект SGBitmapSymbol

### Методы:

LoadSymbol(nSymbol)

Эта функция загружает графические данные нового символа из редактора. При этом старые данные удаляются. Функция может быть использована для сохранения памяти интерпретатора скрипта, если в скрипте нет операторов явного удаления объекта.

В принципе, при создании нового объекта символа с помощью функции GetSymbolObject(…)SGBitmapFontServer\_GetSymbolObject данные автоматически копируются в него. Однако с помощью этой функции один и тот же объект может быть использован повторно.

[[12]](#footnote-12)#Объект SGBitmapSymbol

### Методы:

StoreSymbol(nSymbol)

Эта функция копирует графические данные объекта в сам редактор. Собственно к экспорту она никакого отношения не имеет, однако с помощью неё могут быть выполнены простейшие действия по редактированию символов в редакторе. Если, к примеру необходимо повернуть все символы по часовой стрелке на 90 градусов, можно запустить следующий скрипт:

 var Srv = WScript.CreateObject("SGFED.SGBitmapFontServer");

 var MySymbol = Srv.GetSymbolObject(0);

 for(Count = 0; Count < Srv.SymbolsCount; Count++)

 {

 MySymbol.LoadSymbol(Count);

 MySymbol.Rotate();

 MySymbol.StoreSymbol(Count);

 }

[[13]](#footnote-13)#Объект SGBitmapSymbol

### Методы:

InsertLines(StartLine, LinesCount)

Эта функция добавляет чистые горизонтальные линии, начиная с указанной в StartLine. Всего добавляется «LinesCount» линий. Все данные, что идут ниже – сдвигаются.

[[14]](#footnote-14)#Объект SGBitmapSymbol

### Методы:

GetPictureBegin()

Эта функция сканирует графические данные символа слева направо и возвращает номер первого столбца, в котором есть хотя-бы один установленный бит. Если символ чист, она возвращает 0. Её можно использовать вместе с функцией GetPictureEnd(…)SGBitmapSymbol\_GetPictureEnd для экспортирования шрифтов с переменной шириной символов.

[[15]](#footnote-15)#Объект SGBitmapSymbol

### Методы:

GetPictureEnd()

Эта функция сканирует графические данные символа слева направо и возвращает номер первого столбца, в котором нет ни одного установленного бита, после столбцов, в которых был хотя-бы один установленный бит. Если символ чист, она возвращает то же значение, что и свойство WidthSGBitmapSymbol\_Properties. Её можно использовать вместе с функцией GetPictureBegin(…)SGBitmapSymbol\_GetPictureBegin для экспортирования шрифтов с переменной шириной символов.

[[16]](#footnote-16)#Объект SGBitmapSymbol

### Методы: (вер. 2)

Scroll(Left, Top, Width, Height, Count, bHorz)

Эта функция скроллирует графические данные в заданном прямоугольнике на число бит, заданное переменной Count. Если последний булевый параметр не равен нулю, то происходит скроллирование слева на право. Иначе – сверху вниз.

[[17]](#footnote-17)#Объект SGBitmapSymbol

### Свойства:

Width

Текущая ширина символа, хранящегося в объекте, в пикселах. Так как данные по ширине всегда выровнены на границу байта, значение этого свойства всегда кратно 8.

Height

Текущая высота символа, хранящегося в объекте, в пикселах.

DataBytes(Index)

Предоставляет доступ для записи и чтения непосредственно байтов графической информации по указанному индексу из буфера объекта.

DataBytesStr(Index, Mode)

Предоставляет доступ для чтения непосредственно байтов графической информации по указанному индексу из буфера объекта. Данные конвертируются в текстовый формат. Если Mode = 0, используется десятичный формат. Mode = 1 – шестнадцатеричный формат с выравниванием к ширине в 2 символа.

DataWords(Index)

Предоставляет доступ для записи и чтения непосредственно слов графической информации по указанному индексу из буфера объекта. В буфере старший байт в слове всегда идёт за младшим. Если размер буфера равен нечётному числу байт, то старший байт последнего слова в операции чтения равен 0, а в операции записи игнорируется.

DataWordsStr(Index, Mode)

Предоставляет доступ для чтения непосредственно слов графической информации по указанному индексу из буфера объекта. Данные конвертируются в текстовый формат. Если Mode = 0, используется десятичный формат. Mode = 1 – шестнадцатеричный формат с выравниванием к ширине в 4 символа.

DataPixels(X, Y)

Предоставляет доступ для записи и чтения непосредственно битов графической информации символа. Бит указывается его смещением по координатам X и Y.

1. # first\_steps [↑](#footnote-ref-1)
2. # SGBitmapFontServer\_GetSymbolObject [↑](#footnote-ref-2)
3. # SGBitmapFontServer\_ShowMessage [↑](#footnote-ref-3)
4. # SGBitmapFontServer\_SaveUndoState [↑](#footnote-ref-4)
5. # SGBitmapFontServer\_Properties [↑](#footnote-ref-5)
6. # SGBitmapSymbol\_CopyRect [↑](#footnote-ref-6)
7. # SGBitmapSymbol\_ClearRect [↑](#footnote-ref-7)
8. # SGBitmapSymbol\_Rotate [↑](#footnote-ref-8)
9. # SGBitmapSymbol\_RemoveLines [↑](#footnote-ref-9)
10. # SGBitmapSymbol\_MirrorRect [↑](#footnote-ref-10)
11. # SGBitmapSymbol\_LoadSymbol [↑](#footnote-ref-11)
12. # SGBitmapSymbol\_StoreSymbol [↑](#footnote-ref-12)
13. # SGBitmapSymbol\_InsertLines [↑](#footnote-ref-13)
14. # SGBitmapSymbol\_GetPictureBegin [↑](#footnote-ref-14)
15. # SGBitmapSymbol\_GetPictureEnd [↑](#footnote-ref-15)
16. # SGBitmapSymbol\_Scroll [↑](#footnote-ref-16)
17. # SGBitmapSymbol\_Properties [↑](#footnote-ref-17)