УДК 004.924

ИНТЕРПРЕТАТОР СКРИПТОВ ДЛЯ ГРАФИЧЕСКОГО РЕДАКТОРА

В.В. Карабчевский, С.Н.Магдалина
Донецкий национальный технический университет
karabch@pmi.dgtu.donetsk.ua

В статье рассматривается интерпретатор скриптов, который может использоваться как часть более крупной системы для расширения ее возможностей. Интерпретатор разработан для графического редактора растрово-векторного типа для упрощения процесса генерации изображений, однако, может быть интегрирован и в другие системы.

Введение

При разработке крупных систем программисты стараются учесть все потребности потенциальных пользователей программы и оснащают ее набором необходимых инструментов. Однако, часто так бывает, что возможностей программы не хватает для удовлетворения потребностей пользователей. Существующими в программе средствами бывает сложно или даже невозможно осуществить цели пользователей. В таких ситуациях разработчикам приходится выпускать различные дополнения, патчи и новые версии продукта.

Однако, такая схема не обеспечивает приемлемой мобильности – сколько бы ни выходило дополнений к программе, потребности всех пользователей все равно не будут удовлетворены. Кроме того, разработка дополнений лежит на плечах узкого круга разработчиков системы, что затрудняет выход новых версий системы.

В связи с этим, в рамках проекта по разработке графического редактора [1] был разработан интерпретатор скриптов, который помог бы расширить базовые возможности графического редактора, а также был бы полезен при разработке других систем.

Структура интерпретатора и его интеграция с другими программами

Интерпретатор скриптов KShScript представляет собой библиотеку, написанную на языке C++ в среде Visual Studio 2010 [2].

Интерпретатор предназначен для того, чтобы дать возможность конечным пользователям программы влиять на ее возможности без

изменения исходного кода программы. Однако, пользователь сможет воспользоваться возможностями интерпретатора только если программист, который пишет программу, интегрирует библиотеку в программу. Распределение ролей программиста и пользователя при работе с интерпретатором показано на рисунке 1.

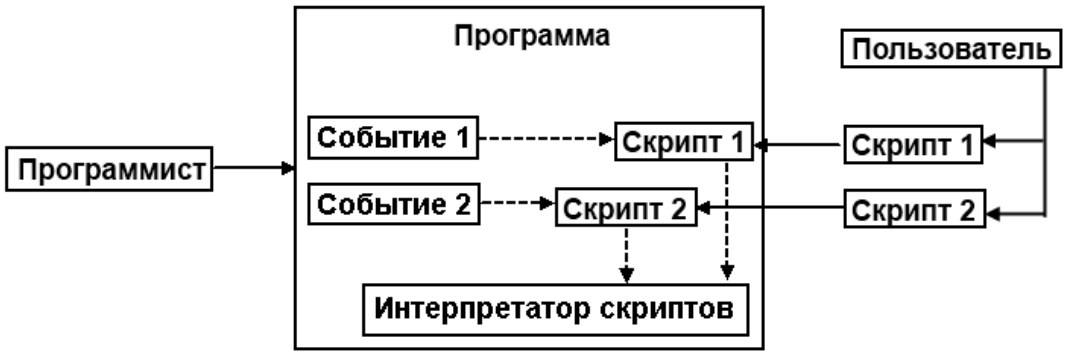

Рисунок 1 – Распределение ролей при работе с интерпретатором.

Скрипты могут вызываться разными методами, однако, наиболее логично применять для этого событийную модель.

Программист пишет программу, в которой наступают те или иные события. Пользователь, в свою очередь, пишет скрипты для интерпретатора скриптов, который встроен в программу, и связывает эти скрипты с событиями. После этого каждый раз при наступлении того или иного события, будет выполняться связанный с ним скрипт, который будет воздействовать на данные программы.

Применение интерпретатора в графике

Основная функция интерпретатора скриптов – расширение возможностей программ, в которые он интегрирован [3]. Эта проблема является актуальной в среде графических редакторов. Для них постоянно пишут плагины, выходят новые версии и пакеты – все это для того, чтобы удовлетворить растущие требования пользователей. Использование интерпретатора скриптов позволяет проводить адаптацию графических редакторов под нужды пользователей быстрее.

Рассмотрим применение интерпретатора для расширения базовых возможностей программы на примере графического редактора KShPainter. Редактор написан на языке C++ в среде Visual Studio 2010 с применением технологии DirectX [4]. Окно открытого приложения представлено на рисунке 2.

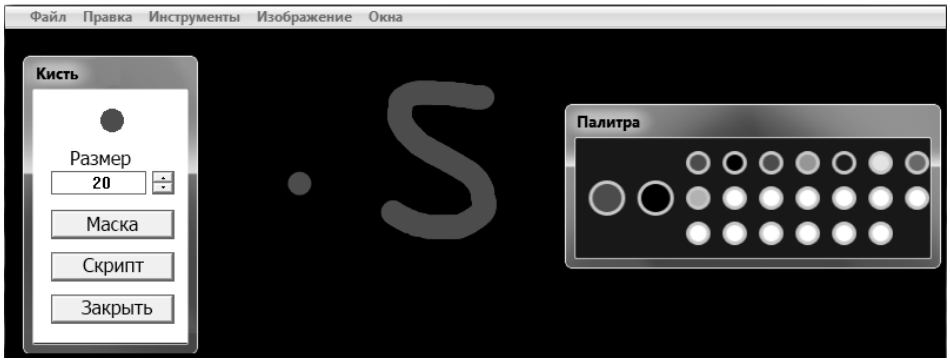

Рисунок 2 – Окно программы KShPainter.

Графический редактор KShPainter предоставляет возможность ручного изменения изображения с помощью инструмента "Кисть". Стандартная форма кисти представляет собой круг. Однако, часто для достижения того или иного художественного эффекта необходимо применять кисти различных форм.

Для того, чтобы изменить форму кисти, можно загрузить пользовательский скрипт. Одним из наиболее изящных и красивых видов кистей является каллиграфическая кисть. Текст скрипта, который заменяет стандартную круглую кисть на каллиграфическую, представлен на рисунке 3.

```
PR_VAR A PT_ABYTE image
PR_VAR A PT_INT  brushSize
PR_VAR A PT_INT  imageSizeX
PR_VAR A PT_INT  imageSizeY
PR_VAR A PT_INT  pointX
PR_VAR A PT_INT  pointY
PR_VAR A PT_BYTE  colorR
PR_VAR A PT_BYTE  colorG
PR_VAR A PT_BYTE  colorB
PR_VAR A PT_BYTE  colorA
PR_VAR L PT_INT  index
PR_VAR L PT_INT  i
PR_VAR L PT_INT  j
PR_PROGRAM
i = pointY - brushSize
while (i < pointY + brushSize)
{
     j = pointX - brushSize
     while (j < pointX + brushSize)
     {
          if ((j < imageSizeX) & (j > -1))
          {
               if ((i < imageSizeY) & (i > -1))
               {
                    index = j + i * imageSizeX
                    image[index] = colorR
                    index = index + 1
                    image[index] = colorG
                    index = index + 1
                    image[index] = colorB
                    index = index + 1
                    image[index] = colorA
               }
          }
     j = j + 4
     }
i = i + 1
}
PR_END
```

Рисунок 3 – Текст скрипта рисования каллиграфической кистью.

Детальное рассмотрение текста скрипта выходит за рамки данной статьи. Однако, стоит отметить, что в скрипт из внешней программы (графического редактора в нашем случае) передаются : массив байт поверхности изображения (image), размер кисти, которой

выполняется рисование (brushSize), размеры изображения в пикселях (imageSizeX, imageSizeY), точка рисования (pointX, pointY) и цвет кисти (color). Остальные переменные являются локальными для скрипта и не влияют на значения переменных программы непосредственно.

В результате работы скрипта кисть вместо круглой формы приобретает форму линии. Результат применения измененной кисти представлен на рисунке 4.

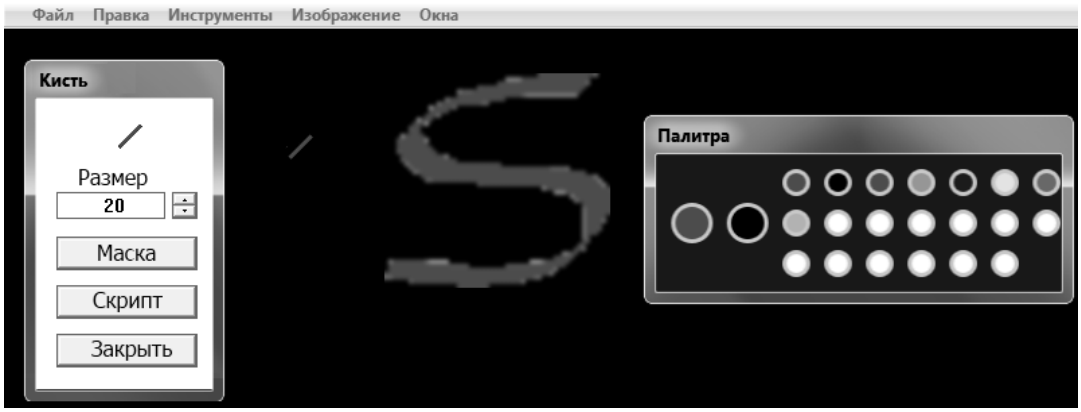

Рисунок 4 – Демонстрация работы скрипта.

Выводы

Приведенный пример позволяет наглядно показать функции интерпретатора скриптов при расширении возможностей программ.

Использование интерпретатора скриптов в графическом редакторе KShPainter реализовано во множестве инструментов и функций, что позволяет легко произвести адаптацию редактора под нужды конкретных пользователей, а также расширить базовые возможности редактора.

Список литературы

1. Карабчевский В.В., Магдалина С.Н. Визуальное создание двумерных текстур средствами DirectX 9.0c // Наукові праці Донецького національного технічного університету. Серія "Інформатика, кібернетика та обчислювальна техніка". Випуск 10 (153). – Донецьк: ДонНТУ. – 2009. – С. 167–171.

2. http://msdn.microsoft.com – документация по MFC.

3. Секунов Н.Ю. Самоучитель Visual C++. СПб.: БХВ-Петербург, 2002. – 250 с.: ил.

4. Станислав Горнаков. DirectX 9: Уроки программирования на C++. СПб.: БХВ – Петербург, 2005. – 400 с.: ил.

Получено 07.09.2011