НАСТРОЙКА СРЕДЫ nanoCAD ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Рано или поздно у опытного пользователя появляется необходимость расширить штатный функционал САПР своими скриптами и командами, автоматизирующими каждодневную рутину. Мы регулярно получаем запросы на тему "Как создать свое меню?", "Как зарегистрировать свою команду?", "Как прописать скрипт в среде nanoCAD?". В этой статье мы решили объединить типовые вопросы и подробно продемонстрировать на несложном примере, как пользователь может настроить платформу nanoCAD под себя и сделать ее чуть более функциональной.

В качестве примера возьмем задачку по организации библиотеки *.dwg-файлов – у каждого пользователя за время работы накапливается достаточно материалов, которые в дальнейшем используются как типовые решения: элементы из старых чертежей, базы блоков и т.д. Мы рассмотрим один из возможных вариантов организации собственной библиотеки блоков в среде nanoCAD и параллельно покажем, как зарегистрировать произвольную скрипт-функцию в меню и командной строке. По аналогии вы сможете написать более сложные скрипты и автоматизировать работу по другим направлениям (например, по расчетам, графическим построениям, по связи с базой данных и т.п.).

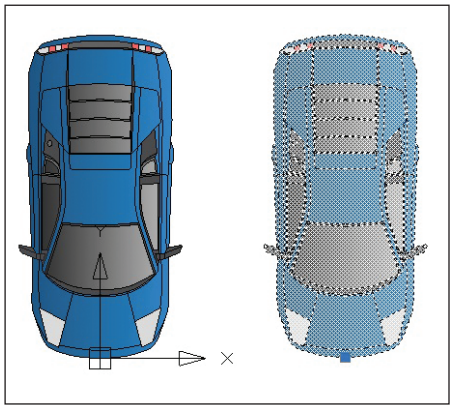

Рис. 1. Вставляемые блоки лучше всего вычертить в нулевых координатах, чтобы была понятна точка вставки блока

Эта статья ориентирована в первую очередь на продвинутых пользователей, системных администраторов и специалистов, не боящихся программирования, – тех, кто уже сейчас хочет автоматизировать и упростить свою работу.

Вступление: подготовка .*dwg-файлов

Поскольку в нашем примере мы работаем с отдельными *.dwg-файлами, давайте чуть-чуть проговорим о том, как эти блоки лучше подготовить и где хранить. Понятно, что вы можете использовать собственный набор блоков, а мы в качестве примера будем использовать два *.dwg-файла: BlueCar.dwg и GreenCar.dwg.

Графику блока мы вычертили в нулевых координатах, так как именно нулевая точка будет базовой (точкой вставки) будущего блока. Поэтому ваш блок лучше начинать вычерчивать с координат x=0, y=0. Если блок уже вычерчен, просто размещаем его так, чтобы в координатах 0,0 располагалась удобная с вашей точки зрения точка вставки. См. на рис. 1: слева – расположение автомобиля в файле BlueCar.dwg, справа – блок после вставки с базовой точкой.

Также мы используем правило: один блок – один *.dwg-файл. Это позволит создать на каждый блок отдельную команду вставки.

Далее собираем все блоки в папке *MyBlocks*, которую либо сохраняем локально на своем диске (мы, например, будем использовать путь *C:/MyBlocks/*), либо размещаем в локальной сети. В последнем случае библиотекой смогут воспользоваться и ваши коллеги (то есть путь будет примернотаким:*//MYSERVER/MyBlocks/*). Ок, теперь мы готовы оборачивать все это кодом...

Создание команд для вставки блоков

Простейший скрипт для вставки блока, написанный на Visual Basic, представлен ниже – нам кажется, что его текст не требует подробных разъяснений. Он просто размещает блок BlueCar.dwg в текущий чертеж – в бездиалоговом режиме по координатам, указанным пользователем:

```
Dim ms
Dim ut
Dim ptInsert
Set ms = ThisDrawing.ModelSpace
REM доступ к пространству Модели текущего документа
Set ut = ThisDrawing.Utility REM доступ к командной строке текущего документа
ptInsert = ut.GetPoint("0,0,0", "Укажите точку вставки")          REM запрос к пользователю координат точки вставки блока
ms.InsertBlock ptInsert," C:\MyBlocks\GreenCar.dwg",1,1,1,0        REM вставляем блок в пространство модели
```

Откуда мы это знаем? Все это описано в стандартном SDK (Software Developer Kit) к nanoCAD; приведенные в скрипте команды – это стандартные API-функции САПР, основанных на *.dwg (рис. 2). Скачать последнюю версию SDK можно на *developer.nanocad.ru*. Там же можно получить примеры скриптов, выполняющих другие интересные задачи в nanoCAD.

Теперь скрипту по вставке блока надо присвоить имя-команду (чтобы вызывать этот скрипт из командной строки nanoCAD). И сделать это надо для каждого блока: один блок – одна команда. Для этого упакуем наш скрипт в специальный xml, – он зарегистрирует в nanoCAD две новые команды, *BlueCar* и *GreenCar*, которые соответственно вставляют *BlueCar.dwg* и *GreenCar.dwg*. Возможно, это не самое рациональное решение с точки зрения "правильности" оформления кода, но мы же сейчас не оптимизацией занимаемся, правда? В итоге получаем следующее:

```
<?xml version="1.0" encoding="utf-8"?>
<package>
  <command name="BlueCar"
  weight="30" cmdtype="1">
    <script lang="VBS"><![CDATA[
      Dim ms
      Dim ut
      Dim ptInsert
      Set ms = ThisDrawing.
      ModelSpace
      Set ut = ThisDrawing.Utility
      ptInsert = ut.GetPoint("0,0,0",
      "Укажите точку вставки")
      ms.InsertBlock ptInsert,"C:\
      MyBlocks\BlueCar.dwg",1,1,1,0
    ]]></script>
  </command>
  <command name="GreenCar"
  weight="30" cmdtype="1">
    <script lang="VBS"><![CDATA[
      Dim ms
      Dim ut
      Dim ptInsert
      Set ms = ThisDrawing.
      ModelSpace
      Set ut = ThisDrawing.Utility
      ptInsert = ut.GetPoint("0,0,0",
      "Укажите точку вставки")
      ms.InsertBlock ptInsert,"C:\
      MyBlocks\GreenCar.dwg",1,1,1,0
    ]]></script>
  </command>
</package>
```

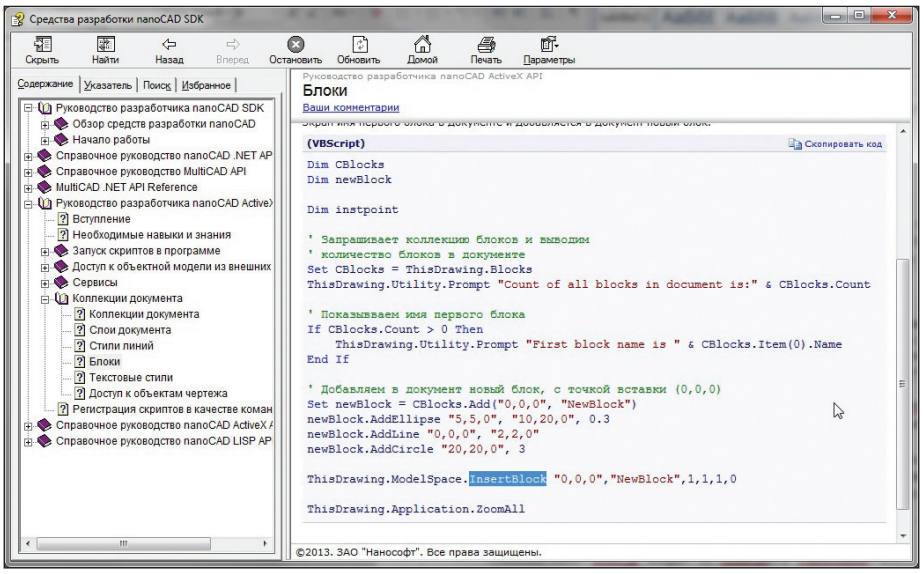

Рис. 2. Документация разработчика nanoCAD подробно описывает все необходимые функции для работы со средой nanoCAD посредством языков C++, .NET, Visual Basic Script, Java Script и LISP

Вроде данный скрипт также не требует пояснений – xml определяет имя команды, а исполняемый код мы помещаем в теги <script>. Сейчас мы не будем подробно расписывать другие параметры *.xml-файла – если вас интересуют подробности, то опять же обратитесь в SDK к платформе nanoCAD (см. Руководство разработчика nanoCAD ActiveX API, раздел "Регистрация скриптов в качестве команд").Сохраняем xml под именем *MyBlocks.nsf* и будем использовать его при загрузке nanoCAD. Обращаем ваше внимание, что кодировка сохраненного файла должна совпадать с описанием заголовка *.xml-файла. В нашем случае это UTF-8 – не забудьте выставить эту кодировку при сохранении (рис. 3).

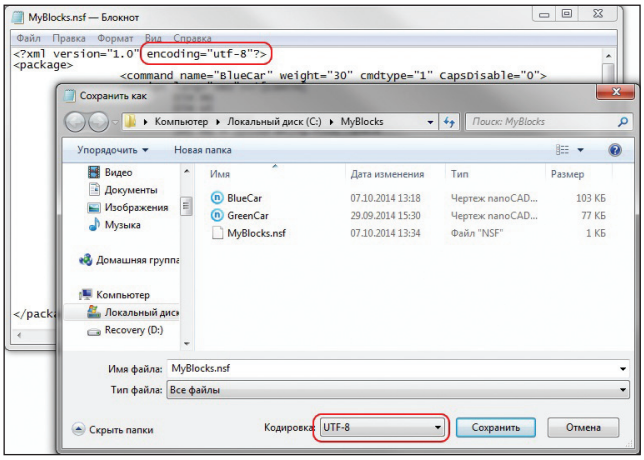

Рис. 3. При сохранении *MyBlocks.nsf* будьте внимательны с кодировками файла – это важно!

Добавление и регистрация команд в командной строке nanoCAD

В простейшем случае для использования новых команд через командную строку файл *MyBlocks.nsf* нужно подгрузить в среду nanoCAD через диалог *Загрузка/выгрузка приложений* из меню *Сервис/Приложения/Загрузка* приложений (рис. 4). И понятно, что файл со скриптами так же, как и блоки, может лежать в локальной сети в общей папке (например, в той же папке с блоками *//MYSERVER/MyBlocks/)*.

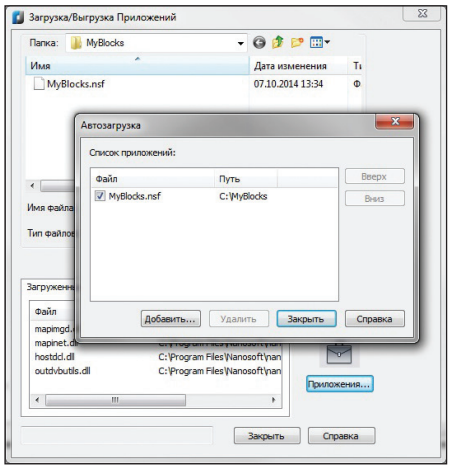

Рис. 4. Загрузка своих команд в среду nanoCAD

Также рекомендуем положить файл *MyBlocks.nsf* в автозагрузку – скрипт будет автоматически загружаться при каждом старте программы. При этом если вы хотите прописать файл MyBlocks.nsf в автозагрузку на каждом рабочем месте пользователя автоматически, то вам нужно с помощью доменных политик инициировать реестр nanoCAD по пути *HKEY_CURRENT_USER/Software/Nanosoft/nanoCADPlus/6.0/Profile/Appload/ Startup*.

А теперь давайте научимся создавать свои пункты меню, горячие клавиши, панели инструментов под наши команды вставки блоков и более тесно интегрируем библиотеку с интерфейсом nanoCAD. Эта интеграция сейчас описывается с помощью *.cfg-файла – обычного текстового файла, имеющего несколько специализированных разделов. Рассмотрим их...

Добавляем описания команды запуска скриптов BlueCar и GreenCar

Для того чтобы наши команды "красиво" описывались в командной строке, строке состояния и всплывающих подсказках, добавим к ним более подробную информацию для nanoCAD:

```
[\configman\commands\BlueCar]
weight=i30  |cmdtype=i0
intername=sBlueCar
DispName=sВставка блока BlueCar
StatusText=sПример вставки блока
BlueCar, основанной на VBS

[\configman\commands\GreenCar]
weight=i30  |cmdtype=i0
intername=sGreenCar
DispName=sВставка блока GreenCar
StatusText=sПример вставки блока
GreenCar, основанной на VBS
```

где DispName – описание команды, отображаемое в командной строке, а StatusText – всплывающий текст-подсказка при наведении курсора на иконку на панели инструментов (рис. 5).

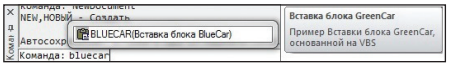

Рис. 5. Все команды, зарегистрированные в nanoCAD, можно "обернуть" дополнительной информацией

Регистрация команд в меню Мои команды

Далее создадим выпадающее меню *Мои команды*, в котором зарегистрируем наши новые команды. Добавим в него подменю с именем *Вставка блоков* и наши команды *BlueCar* и *GreenCar*. Присвоим подпунктам имена – *Вставка BlueCar* и *Вставка GreenCar* соответственно:

```
[\menu\mycommands]
|name=sМои команды

[\menu\mycommands\InsertBlock]
|name=sВставка блоков

[\menu\mycommands\InsertBlock\
BlueCar]          |name=sВставка
BlueCar  |InterName=sBlueCar

[\menu\mycommands\InsertBlock\
GreenCar]         |name=sВставка
GreenCar |intername=sGreenCar
```

Результат – на рис. 6.

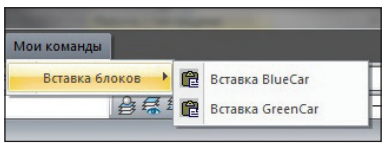

Рис. 6. Добавляем команды в меню

Отображение пользовательской панели инструментов

Отлично, мы добавили выпадающее меню с собственными командами! Давайте по аналогии с меню добавим еще и панель инструментов:

```
[\toolbars\mycommands]
|InitialVisible=f1 |name=sМои команды
[\toolbars\mycommands\BlueCar]
|intername=sBlueCar
[\toolbars\mycommands\GreenCar]
|intername=sGreenCar

[\toolbarspos\mycommands]
|DockPosition=sTop |row=i1 |pos=i2
```

Здесь мы описали свою панель инструментов *Мои команды* с двумя кнопками, включили ее по умолчанию, а также указали, где по умолчанию эта панель будет отображаться в интерфейсе nanoCAD – вверху во втором ряду третьей.

Теперь добавим панель *Мои команды* в список панелей инструментов nanoCAD – в штатное место меню *Вид/Панели* и в контекстное меню:

```
[\menu\View\toolbars\My_toolbars]
|Name=sМои панели
[\menu\View\toolbars\My_toolbars\
ShowToolbar_mycommands]
|Name=sМои команды
|InterName=sShowToolbar_mycommands

[\ToolbarPopupMenu\My_toolbars]
|Name=sМои панели
[\ToolbarPopupMenu\My_toolbars\
ShowToolbar_mycommands]
|Name=sМои команды
|InterName=sShowToolbar_mycommands
```

Плюс нужная функция, которая включает/отключает новую панель инструментов – иначе скроем панель и открыть ее сможем только из диалога Интерфейс:

```
[\configman\commands\sShowToolbar_
MyCommands]
weight=i10  |cmdtype=i0
intername=sShowToolbar_MyCommands
StatusText=sПоказать/скрыть панель
Мои команды
ToolTipText=sПоказать/скрыть панель
Мои команды
DispName=sПоказать/скрыть панель
Мои команды
```

Теперь наша панель полностью прописалась в интерфейсе nanoCAD – для того чтобы увидеть весь список панелей инструментов, нужно зайти в Настройки

интерфейса. А для быстрого включения/отключения нашей новой панели можно зайти в контекстное меню (клик ПКМ на поле панелей инструментов) (рис. 7).

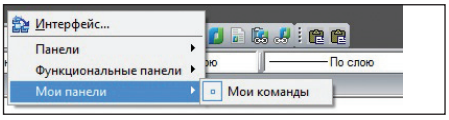

Рис. 7. Управление панелью инструментов

Добавление горячих клавиш

И на финал – для удобства и быстроты использования наших новых команд добавим горячие клавиши. Для синего автомобиля – *Ctrl+B*, для зеленого – *Ctrl+G*:

```
[\Accelerators]
BlueCar=sCtrl+B
GreenCar=sCtrl+G
```

Результат – на рис. 8.

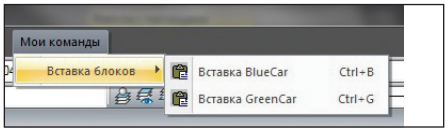

Рис. 8. Назначаем горячие клавиши

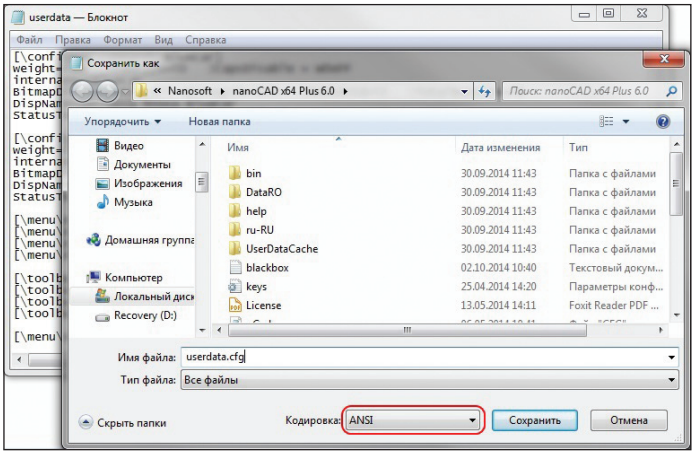

Рис. 9. Сохранение *userdata.cfg* с кодировкой ANSI

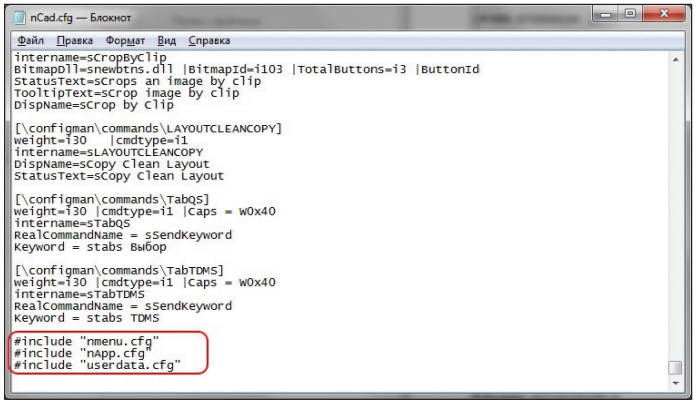

Рис. 10. nanoCAD позволяет подгружать любое количество файлов настроек интерфейса – каждый под свою задачу

Завершаем работы с нашим *.cfg-файлом и сохраняем его под именем *userdata.cfg*. Внимание: при сохранении файла нужно выбрать кодировку ANSI (рис. 9).

Сброс настроек пользовательского интерфейса nanoCAD

Осталось совсем немного – переместить файл userdata.cfg в папку с установленным nanoCAD. Причем очень важно, чтобы файл с настройками интерфейса назывался именно *userdata.cfg*: дело в том, что все настройки стандартного интерфейса nanoCAD хранятся в файле *nCad.cfg* и в конце это файла происходит автоматическая подгрузка пользовательского файла настроек *userdata.cfg* – поэтому в простейшем случае для того чтобы подгрузить настройки пользователя, достаточно положить соответствующий файл в папку к nanoCAD. Если же вы хотите использовать свои файлы, то не забудьте прописать их по аналогии в *nCad.cfg* (рис. 10).

Кстати, для саморазвития рекомендуем изучить файл *nCad.cfg* – там полностью описан интерфейс программы nanoCAD и можно найти все приемы описания под свое приложение.

Напоследок нам надо один раз сбросить настройки интерфейса – nanoCAD заново подгрузит описание интерфейса из нашего файла и пропишет его в реестре Windows под текущего пользователя. Поэтому запускаем nanoCAD, заходим в *Сервис/Настройка интерфейса/Интерфейс* и нажимаем кнопку *Сбросить все* (рис. 11) – при следующем запуске в среде nanoCAD появятся наши пункты меню, панель инструментов и другие настройки.

Заключение

Поздравляем! Теперь вы можете наслаждаться новыми возможностями nanoCAD, избавив себя от рутинной работы (рис. 12). Понятно, что сложность скриптов можно увеличивать, поручая им более серьезные задачи, и таким образом существенно расширять возможности платформы nanoCAD, "затачивая" САПР под свои цели.

Напоследок хотелось бы еще раз подчеркнуть, что это лишь один из способов пользовательской организации интерфейса nanoCAD, ориентированный на тех, кто желает заглянуть вглубь. В ближайшее время мы планируем включить в nanoCAD визуальный редактор, позволяющий работать с интерфейсом более

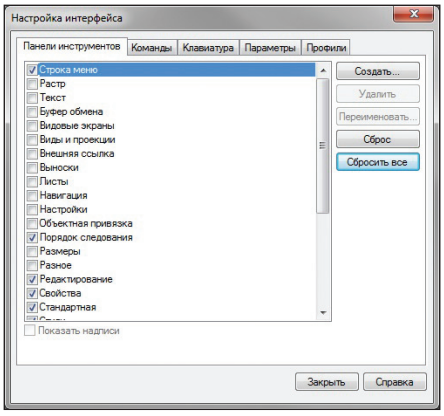

Рис. 11. Для изменения интерфейса надо один раз сбросить настройки nanoCAD через диалог *Настройка интерфейса*

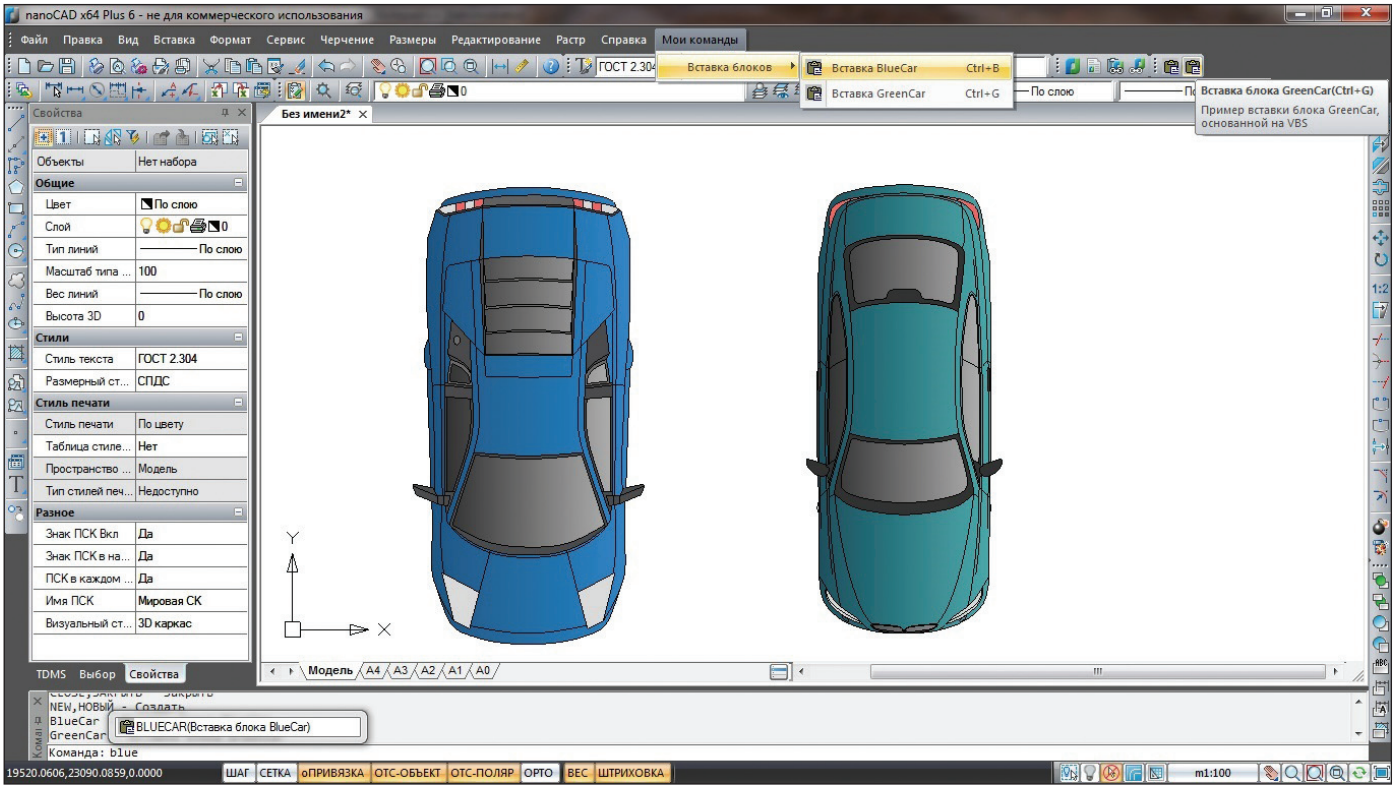

Рис. 12. Пример работающего приложения в среде nanoCAD со своими командами в меню, на панелях инструментов и в командной строке

дружелюбным по отношению к пользователю способом и совместимый как с *.cfg-, так и *.cuix-файлами. Да и для организации библиотек будет более удобный инструмент, аналогичный инструментальным палитрам AutoCAD. Но вариант, приведенный в нашей статье, удобен тем, что позволяет не просто понять "как все устроено", но и выйти за границы штатных возможностей продукта. Плюс к тому можно попробовать автоматизировать распространение настроек nanoCAD с вашими скриптами по рабочим местам пользователей с помощью (например) доменных политик – мы включили в статью несколько намеков на это.

В общем, мы призываем почитать, проанализировать, а затем самостоятельно поэкспериментировать с полученными знаниями. Попытайтесь создать файлы *MyBlocks.nsf* и *userdata.cfg* по инструкциям из нашей статьи или скачайте уже готовые файлы примеров по ссылке *https://corp.nanocad.ru/docs/pub/2ad8e6567299 4b033fc3e305a8c96a02/NC60DemoScripts.rar,* вложите их в папку с установленным nanoCAD, а затем подумайте, как вы можете применить эти знания к своим задачам.

Дополнительно обращаем ваше внимание, что по аналогии вы можете добавить свои настройки в любое вертикальное приложение, построенное на базе nanoCAD. То есть если в диалог *Загрузка/Выгрузка приложений* добавить файл *MyBlocks.nsf*, а файл *userdata.cfg* положить в папку с установленным nanoCAD СПДС или nanoCAD Механика и один раз сбросить интерфейс, то ваши команды появятся и в приложениях!

Удачной автоматизации!

Сергей Спирин,
Денис Ожигин
ЗАО "Нанософт"
Тел.: (495) 645-8626

nanoCAD ОПС – новая база данных

НОВОСТЬ

Доступна новая база данных оборудования НПО "Спектрон" (г. Екатеринбург) для nanoCAD ОПС.

В базу данных включены следующие типы оборудования:

- инфракрасные извещатели пламени в универсальном, промышленном и взрывозащищенном исполнении (Спектрон серия 200);
- ультрафиолетовые извещатели пламени в универсальном, промышленном и взрывозащищенном исполнении (Спектрон серия 400);
- многодиапазонные (ИК/УФ) извещатели пламени в универсальном, промышленном и взрывозащищенном исполнении (Спектрон серия 600);
- тепловые извещатели "ИП-101-Спектрон" во взрывозащищенном исполнении;
- термокожухи серии "Релион" в промышленном, взрывозащищенном и бюджетном исполнении.

Научно-производственное объединение "Спектрон" – ведущий российский производитель высокотехнологичного оборудования для обеспечения безопасности объектов и граждан – создано в 2000 году. Предприятие выпускает широкий спектр продукции в области пожарной и промышленной безопасности, противокриминальной и антитеррористической защиты: пожарные извещатели, приемо-контрольные приборы для охранной и пожарной сигнализации, устройства грозозащиты, изделия для защиты компонентов систем видеонаблюдения, компоненты систем оповещения.

Пожарные извещатели "Спектрон", представленные в базе, совместимы со всеми приемо-контрольными приборами и комплексными системами безопасности российских и зарубежных производителей. Термокожухи "Релион", представленные в базе, предназначены для защиты самых распространенных корпусных стационарных видеокамер российских и зарубежных производителей.

Загрузка базы данных производится со страницы абонемента.

Компания "Нанософт" благодарит НПО "Спектрон" за сотрудничество и предоставление материалов для базы данных.