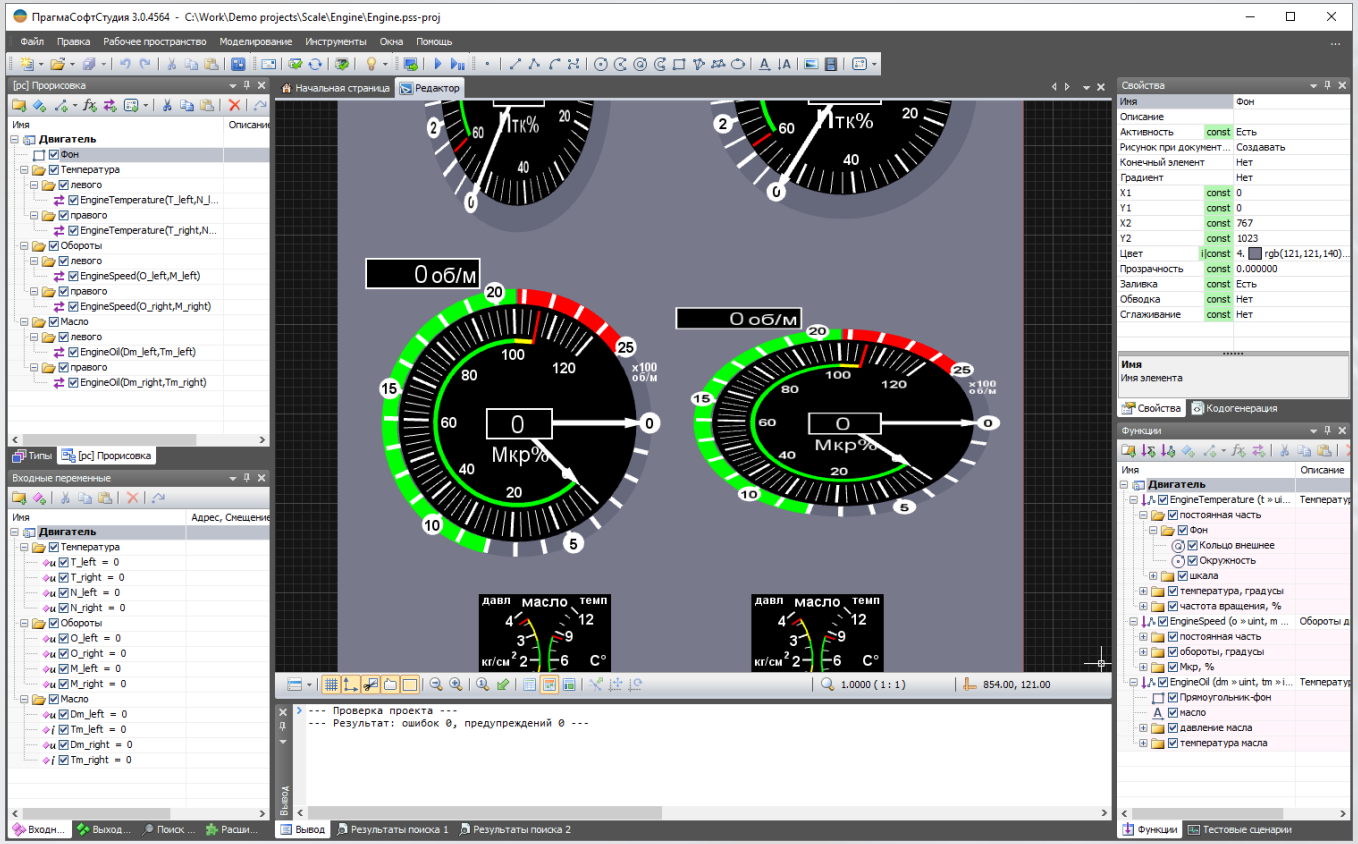
ПрагмаСофтСтудия

Система Автоматизированного ПРоектирования информационно-управляющего поля технических объектов, а также их программного обеспечения



САПР ПрагмаСофтСтудия

Программный комплекс, который поможет Вам как выполнить быстрое прототипирование будущего проекта, так и построить обучающий или исследовательский стенд, а также при необходимости автоматически создаст программный код для развёртывания вашего проекта на аппаратуре.

Прототипирование

Быстрое черновое построение индикационных кадров или их фрагментов.

Построение стендов

- На базе реальных блоков систем индикации с ПО, созданным в САПР
- Путём имитации графики и логики работы систем индикации с помощью САПР

Разработка программного обеспечения

Создание программного кода, отвечающего требованиям стандартов и сертификационных норм. Код создаётся на языке C и способен прорисовывать графику с помощью OpenGL, OpenGL ES и OpenGL SC.

Области применения

Коммерческие, индустриальные и встраиваемые системы, с визуализацией результатов их работы или без неё.

Авиационная промышленность

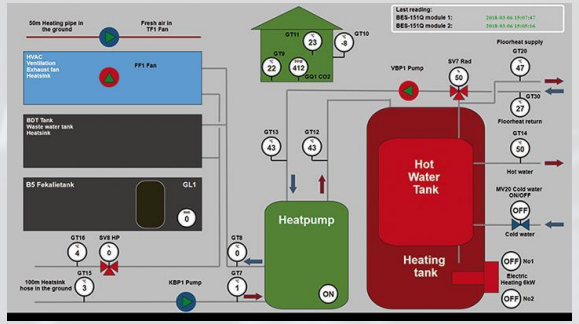
- Информационно-управляющее поле кабины боевых и гражданских самолётов
- Контрольно-проверочная аппаратура и системы технического обслуживания
- Тренажёры и симуляторы
- Стенды поискового, математического и полунатурного моделирования
- Навигационные дисплеи
- Панель приборов современных судов

Кораблестроение

Транспорт и энергетика

- Системы мониторинга состояния транспортных развязок, светофоров, переездов и путей
- Контроль состояния сети и энергоблоков электростанций

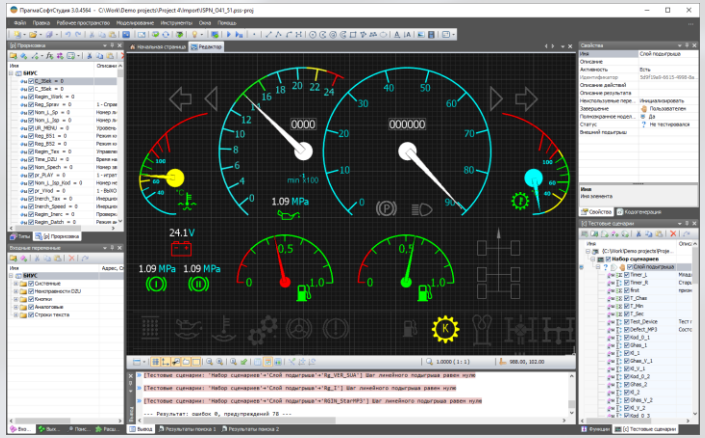
Производство



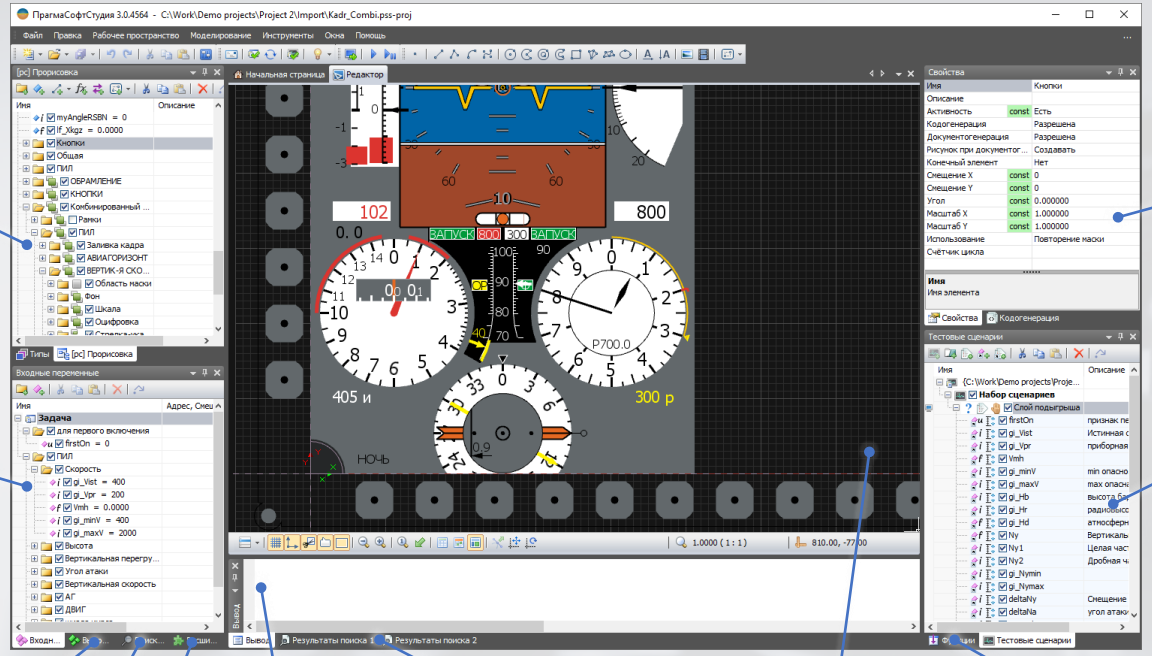
- Оперативный контроль состояния производственных линий и перемещения грузов по цеху
- Системы безопасности, жизнеобеспечения и кондиционирования воздуха
- Мониторы станков с компьютерным управлением

САПР ПрагмаСофтСтудия это:

- Удобный многооконный графический интерфейс
- Механизм ограничения САПР возможностями целевой аппаратуры
- 5 встроенных типов данных, создание из них перечислений и структур, объединение их в массивы
- Многострочные формулы с комментариями и подсветкой синтаксиса
- Более 20 арифметических, логических и побитовых операций доступных при написании формул
- 55 встроенных функций вычислителя САПР, в том числе тригонометрические и функции работы со строками
- 17 графических примитивов, плюс 8 примитивов с обратной связью (кнопок)
- 3 режима маскирования графики
- Использование при построении графики иерархии вложенных друг в друга систем координат
- Возможность выполнения частей проекта в цикле
- Вынесение повторяющейся или библиотечной функциональности в отдельный объект
- Встроенный механизм тестирования и эмуляции входных данных на основе тестовых сценариев
- Эмуляция входных данных проекта с помощью внешних подключаемых dll модулей
- Процесс моделирования для демонстрации созданного проекта в динамике
- Генерация надёжного программного кода на языке С



Графический интерфейс



Основное содержимое проекта. Элементы дерева ссылаются на данные в других окнах программы и связывают их в единый проект.

Входные переменные. Источник внешней информации для проекта. Работают в связке с тестовыми сценариями.

Выходные переменные. Запись результатов выполнения проекта в память компьютера.

Окно поиска входных и выходных переменных в проекте

Окно для работы с внешними плагинами, расширяющими функциональность программы.

Окно информационных сообщений и сообщений об ошибках

Окна результатов поиска входных и выходных переменных в проекте

Графический редактор для создания графики и отображения её в динамике

Окно свойств. Показывает и позволяет изменить параметры выделенных в данный момент элементов в одном из окон программы.

Окно тестирования проекта и эмуляции входной информации для него

Окно функций. Обособленная функциональность доступная для многократного использования в проекте.

Любое из указанных окон может быть перемещено в другое место главного окна или преобразовано в плавающее окно

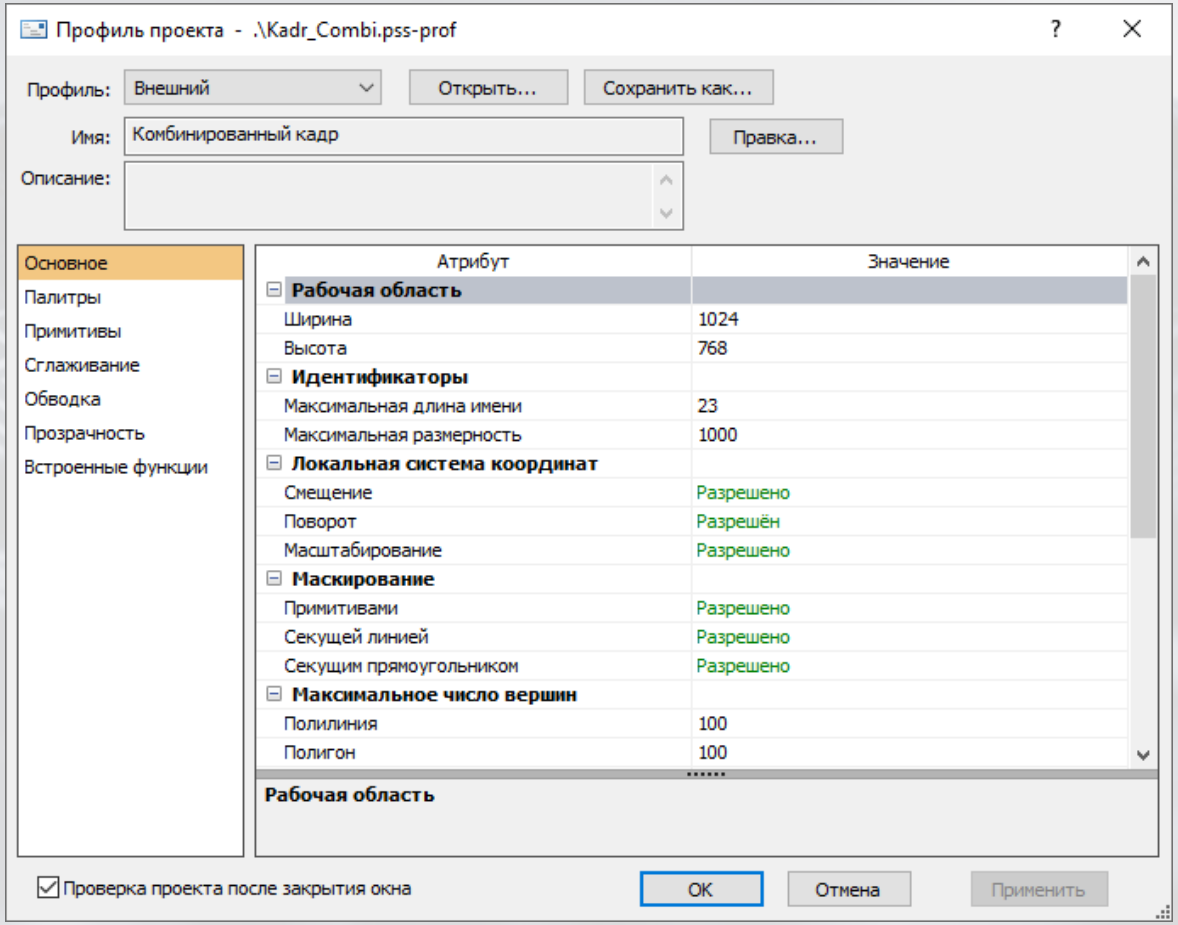
Профиль проекта

Если САПР используется для разработки ПО, то необходимо ограничение его функциональности возможностями аппаратуры, на которой это ПО в последующем будет работать.

Для этого существует Профиль проекта. Это набор параметров и правил, определяющих что и как можно делать в проекте.

Окно Профиля проекта содержит 6 страниц с настройками:

- **Основное**. Базовый набор ограничений функций проекта в разных его областях.
- **Палитры**. Список цветов, толщин и типов линий, а также шрифтов и растровых изображений, которые допустимо использовать при построении графики.
- **Примитивы**. Разрешения на использование примитивов в проекте.
- **Сглаживание**. Определяет возможность сглаживания примитивов при их прорисовке.
- **Обводка**. Разрешает использовать внешнюю обводку примитивов.
- **Встроенные функции**. Содержит список некоторых встроенных функций вычислителя САПР и разрешения на их использование при написании формул.



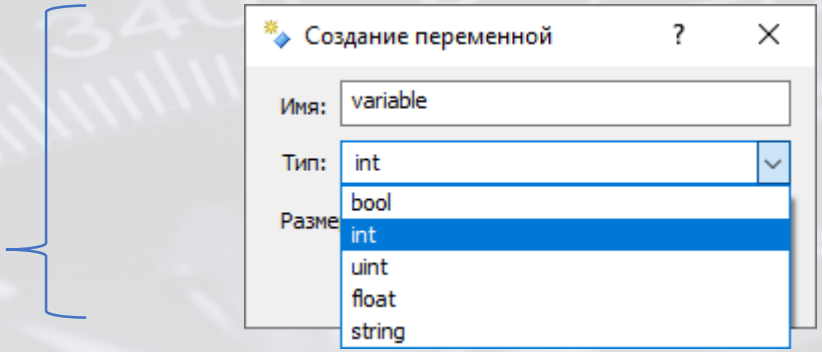
Профиль проекта сохраняется в отдельный файл и может быть использован для разных проектов или нескольких рабочих мест

Типы данных

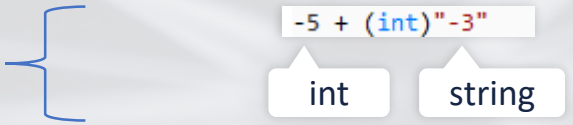
По умолчанию программа предоставляет для использования 5 встроенных типов данных:

bool	логическое значение
int	целое значение со знаком, 32 бита
uint	целое значение без знака, 32 бита
float	вещественное значение, 32 бита
string	строка, последовательность строчных символов

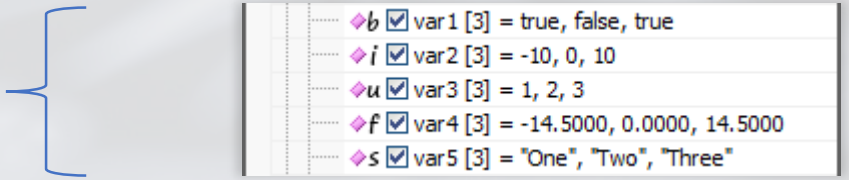
Типы относятся к переменным, создаваемым в проекте, и определяют возможные значения, которые эти переменные могут хранить.



Значения внутри формул и возвращаемый ими результат также относятся к одному из 5 встроенных типов данных.



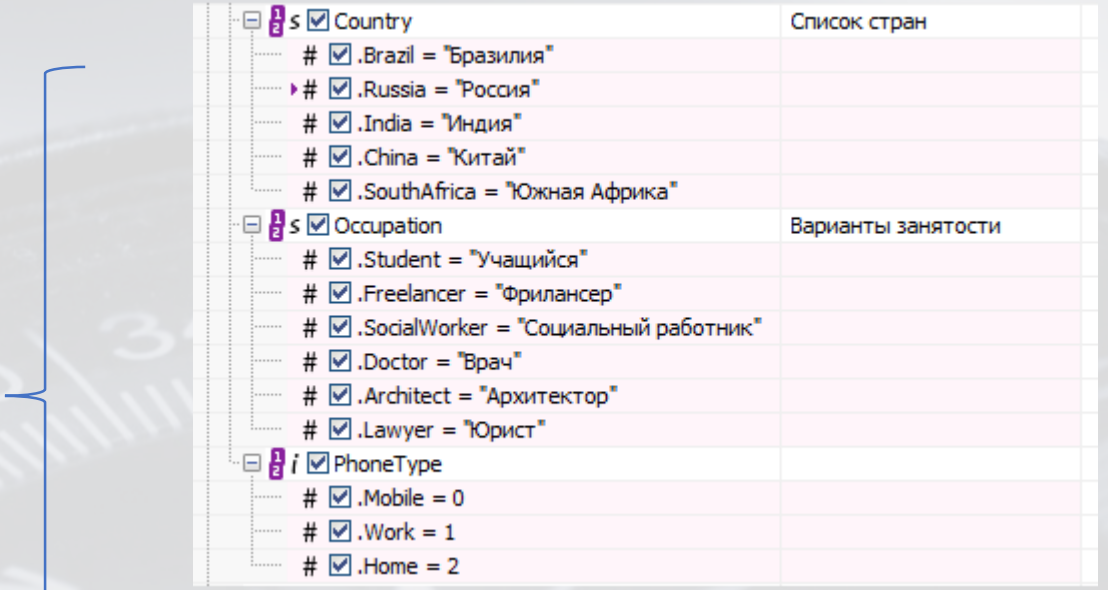
Также переменных могут быть массивами, т.е. хранить более одного значения. Все значения при этом располагаются в памяти последовательно и имеют тип, равный типу переменной.



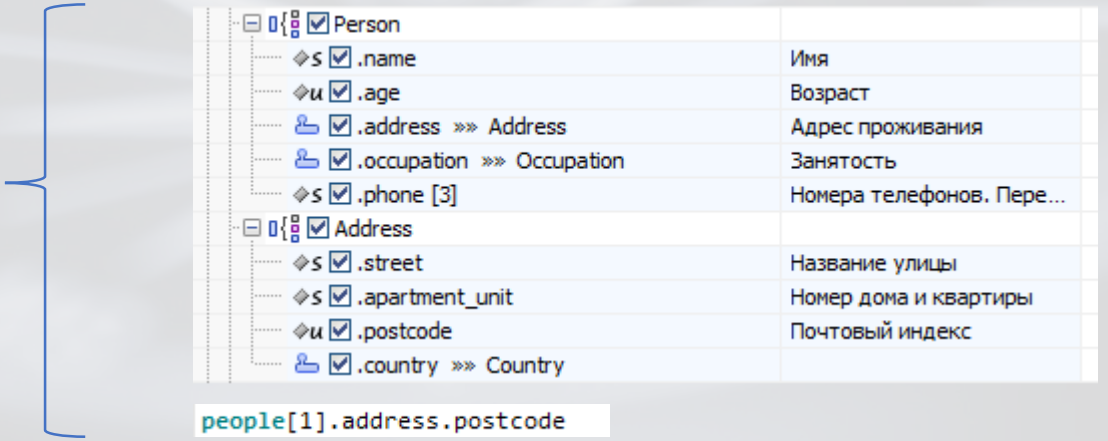
Типы данных

В программе существует возможность дополнять встроенные типы данных (bool, int, uint, float, string) пользовательскими типами, создаваемыми на их основе.

Перечисления. Это именованные списки константных значений. Каждое значение уникально именуется в пределах данного списка. Все значения одного перечисления имеют один из встроенных типов САПР. Значения инициализируются и не могут быть изменены элементами проекта. Доступ к значениям перечисления выполняется с указанием имени перечисления и имени нужного значения разделённых точкой.

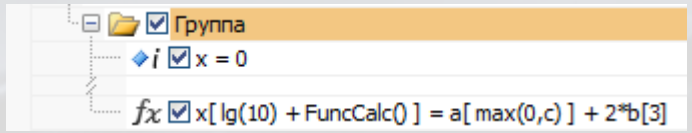


Структуры. Это объединения встроенных типов данных, а также других структур и перечислений в именованные группы. Каждый элемент группы имеет уникальное имя в пределах данной группы, может являться массивом и доступен для изменения элементами проекта. Аналогично переменным встроенных типов, элементы структуры всегда инициализированы. Доступ к ним выполняется с указанием имени переменной, имеющей тип данной структуры, и имени нужного элемента внутри структуры разделённых точкой.



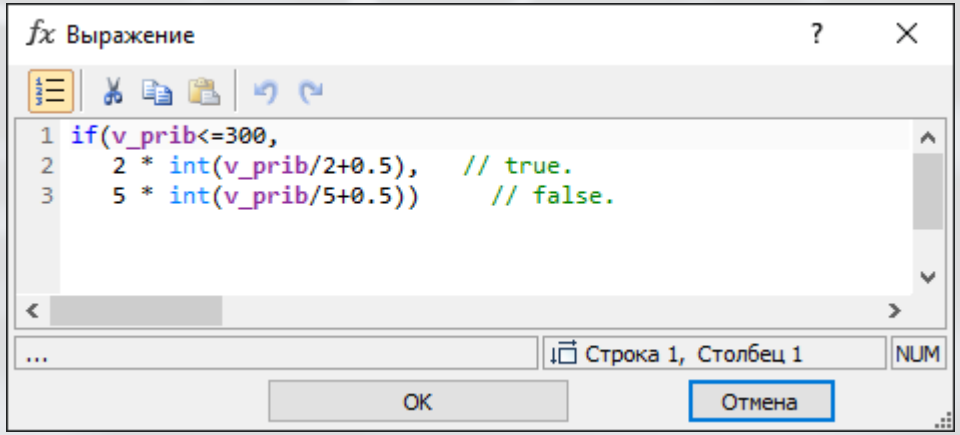
Формулы в программе бывают двух видов:

- В виде отдельных элементов проекта.



- В виде формул в свойствах других элементов проекта. В этом случае формула задаётся в окне Свойства. Левая часть формулы от знака = отсутствует, и результат вычисления формулы приравнивается к значению свойства.

Все окна в САПР, в которых пишутся и редактируются формулы, имеют цветовую подсветку синтаксиса. Формулы могут быть многострочными и иметь комментарии.



Формулы могут содержать:

- Десятичные, шестнадцатеричные и вещественные числа
  ```
  10 + 0xF5 * 0.45
  ```
- Строки
  ```
  "привет мир"
  ```
- Константы и значения перечислений
  ```
  true, false, PI    PhoneType.Home
  ```
- Входные, выходные и локальные переменные
  ```
  var1[ 1 + var2 ] - var3[10]
  ```
- 20 арифметических, логических и побитовых операций
  ```
  2 * 3 && (0xA7C5 << 6)
  ```
- 5 операций приведения типа
  ```
  (int)"12"
  ```
- Вызовы 55 встроенных функций вычислителя
  ```
  sin( ln(10) )
  ```
- Вызовы функций вычисления
  ```
  3.56 + FuncCalc(true,5)
  ```
- Комментарии
  ```
  // comment
  /* comment */
  ```

Вместе с САПР поставляется программа Учебный калькулятор. Она поможет быстро освоить написание формул, выяснить любой нюанс, связанный с синтаксисом, правилом приведения типов и т.д..

Операции

`[]`	обращение к элементу массива по индексу в скобках
`bool int uint float string`	приведение типа
`-`	унарный минус
`! ~`	! - логическое отрицание (инверсия логического значения) ~ - поразрядное НЕ (инверсия всех битов)
`+ -`	сложение, вычитание
`<< >>`	<< - линейный битовый арифметический сдвиг влево >> - линейный битовый арифметический сдвиг вправо
`< <= > >=`	<= - сравнение на меньше или равно >= - сравнение на больше или равно
`== !=`	== - сравнение на равенство != - сравнение на неравенство
`& |`	поразрядное И и ИЛИ
`^`	поразрядное исключающее ИЛИ
`&& ||`	логическое И и ИЛИ

Примитивы, которые можно использовать для создания графики:

Точка

Линия

Полилиния

Дуга

Кривая Безье

Окружность

Сектор круга

Кольцо

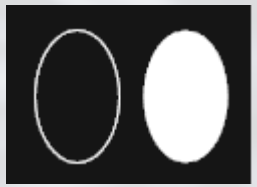
Сектор кольца

Прямоугольник

Веерный полигон

Полосный полигон

Эллипс



Горизонтальный текст

Вертикальный текст

Символ (графический файл)

Видео примитив

Суммарно эти примитивы позволяют воспроизвести практически любую геометрическую фигуру.

Для каждого примитива индивидуально задаётся цвет, толщина, тип линии, наличие сглаживания и обводки, цвет обводки и её толщина, а также наличие заливки. Каждое из этих свойств может быть рассчитано по формуле.

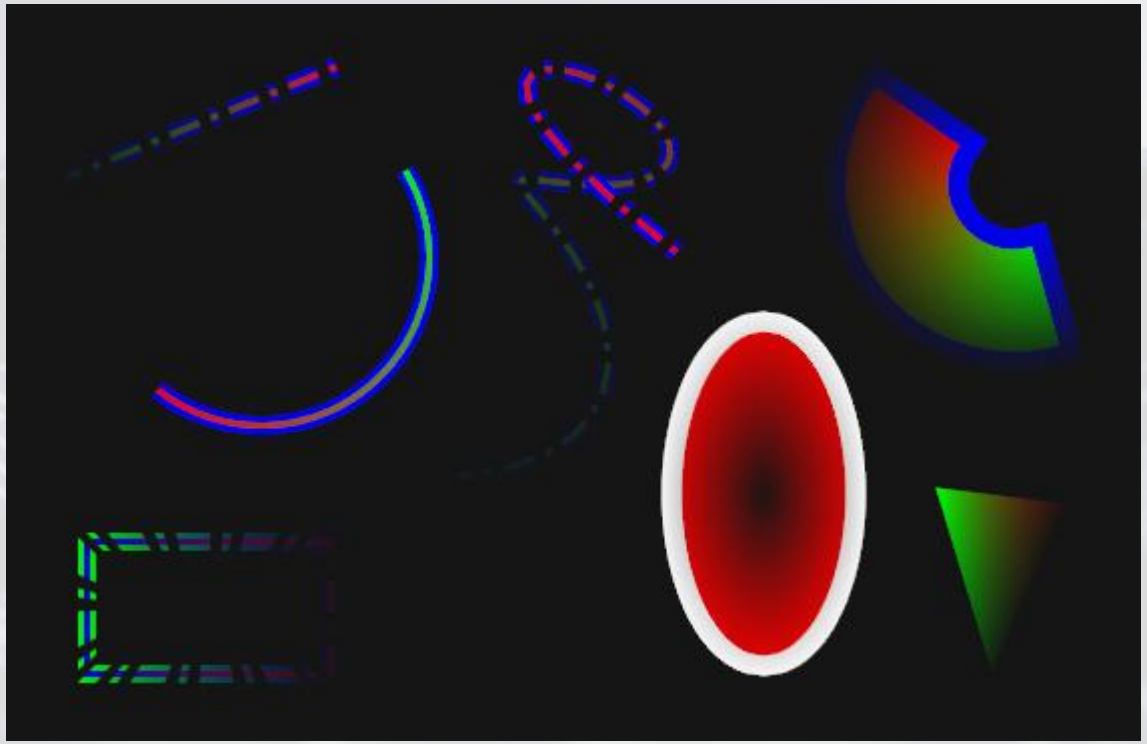
Графические примитивы

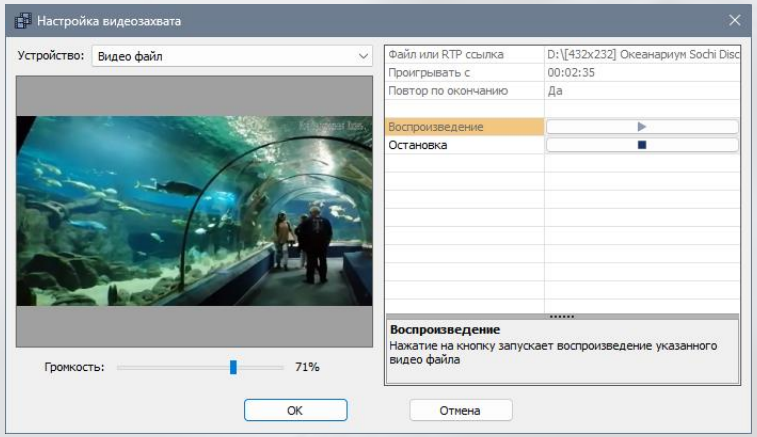
Примитив Символ отображает заданный для него графический файл. Возможные форматы файла: bmp, png и jpg. В случае png файлов они способны содержать прозрачные или полупрозрачные области. Также при необходимости программа даёт возможность запретить показ некоторого цвета в изображении примитива Символ.

Все графические примитивы могут быть прорисованы с использованием прозрачности. При желании, для большинства из них прозрачность задаётся индивидуально для каждой вершины и плавно распределяется между ними, создавая градиент прозрачности.

Также для большинства графических примитивов возможно задание цвета индивидуально для каждой вершины с образованием градиента цвета.

Оба градиента в примитиве могут существовать одновременно и быть заданы формулой.





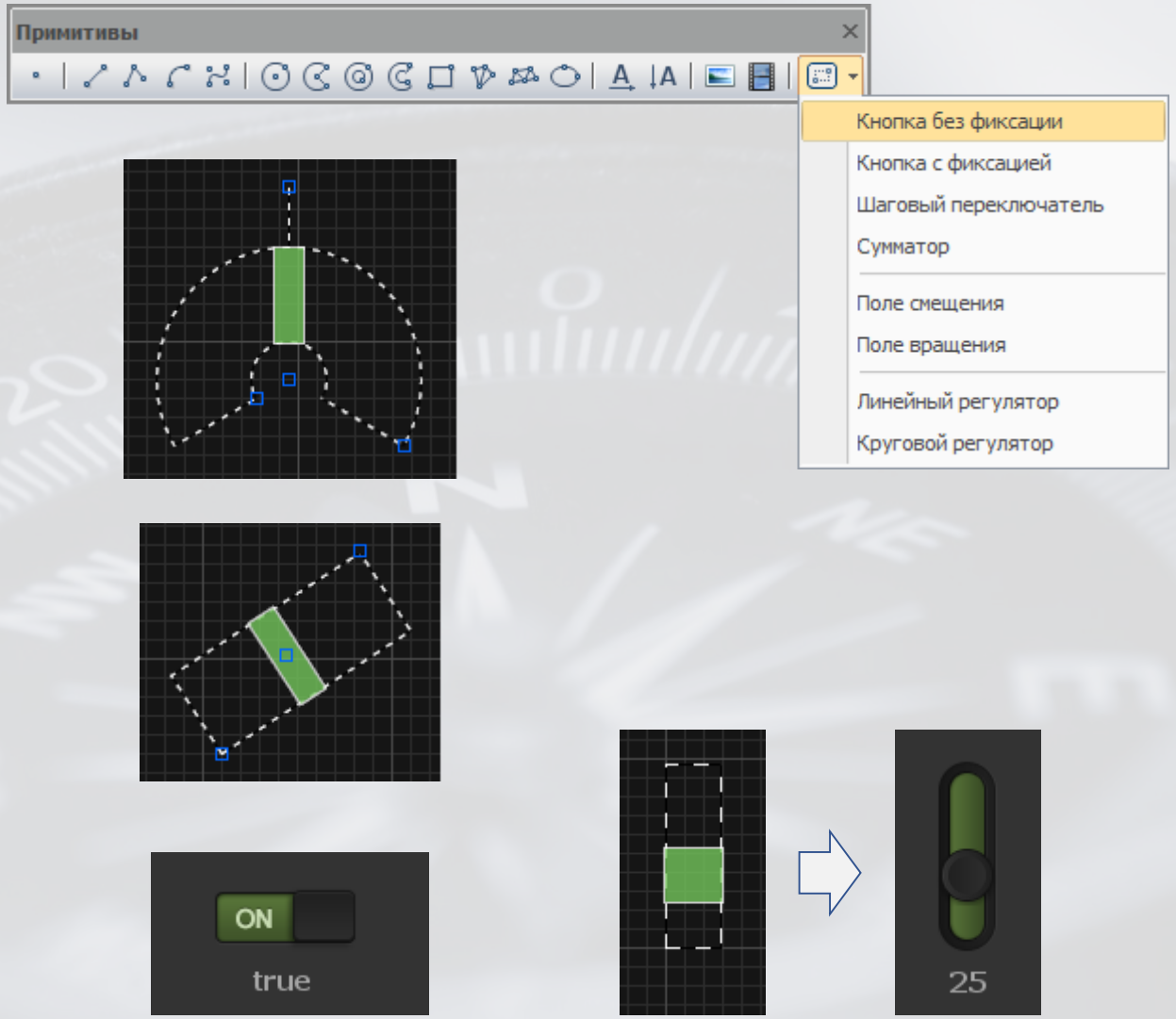
Примитив Видео - это прямоугольный графический примитив, отображающий видео в процессе моделирования САПР. Возможен показ видео из внешнего устройства видеозахвата, веб-камеры, из видео файла или видео передаваемого по сети с использованием протокола RTP.

Кнопки

Кроме графических примитивов существует 8 примитивов, которые отображаются только в режиме редактирования проекта. Это так называемые кнопки, примитивы с обратной связью.

Кнопки используются при отработке проекта в динамике. При совершении в их области каких-либо действий с помощью мыши или сенсорного монитора, информация об этих действиях записывается в связанные с кнопками переменные. Далее, эти переменные, как источник информации могут участвовать в логике и прорисовке проекта.

Поверх чувствительной области кнопки может быть нанесено любое изображение, меняющееся вслед за изменением состояния кнопки, имитируя орган управления.



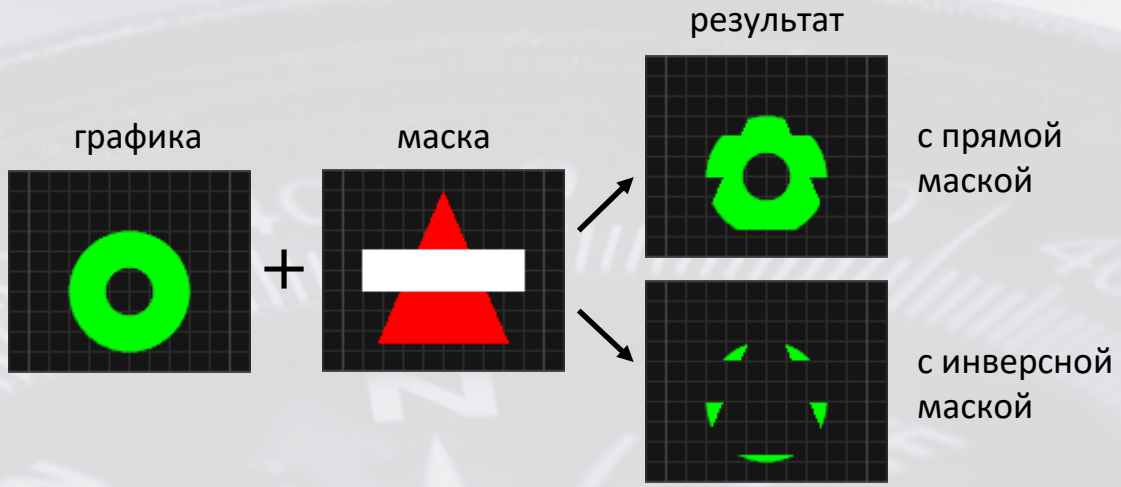
Кнопки применяются одним из двух способов:

- В виде отладочных элементов. Нажимая на них, пользователь изменяет поведение проекта в динамике, меняет его параметры отработки.
- Для создания органов управления. В проектах предназначенных для построения стендов или проведения эргономических исследований на базе кнопок могут быть созданы переключатели и всевозможные регуляторы в том числе имитирующие реально существующие образцы.

Маскирование позволяет отсечь часть графики при её прорисовке. Для этого может быть выбран один из 3 встроенных механизмов, указанных ниже.

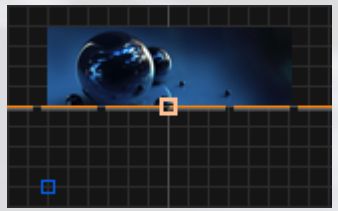
Слой маски

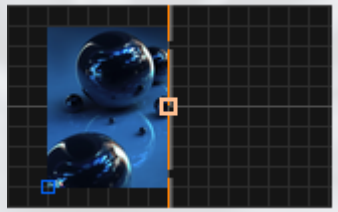
Слой маски — это область, которая не отображается, но используется при прорисовке графики. Маска создаётся телами графических примитивов, т.е. они прорисовываются не на экран, а в слой маски, при этом дополняя или стирая его. Это даёт возможность создать маску практически любой конфигурации, а затем использовать её для скрытия или наоборот показа части графики.



Секущая линия

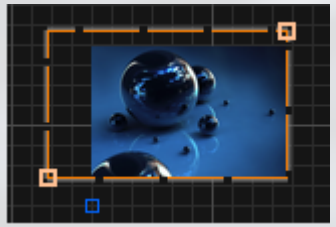
Позволяет задать горизонтальную или вертикальную линию, с одной стороны от которой не будет отображаться графика.





Секущий прямоугольник

Позволяет задать прямоугольное окно, только внутри которого будет видна графика. Всё, что окажется за его приделами, не будет отображаться.



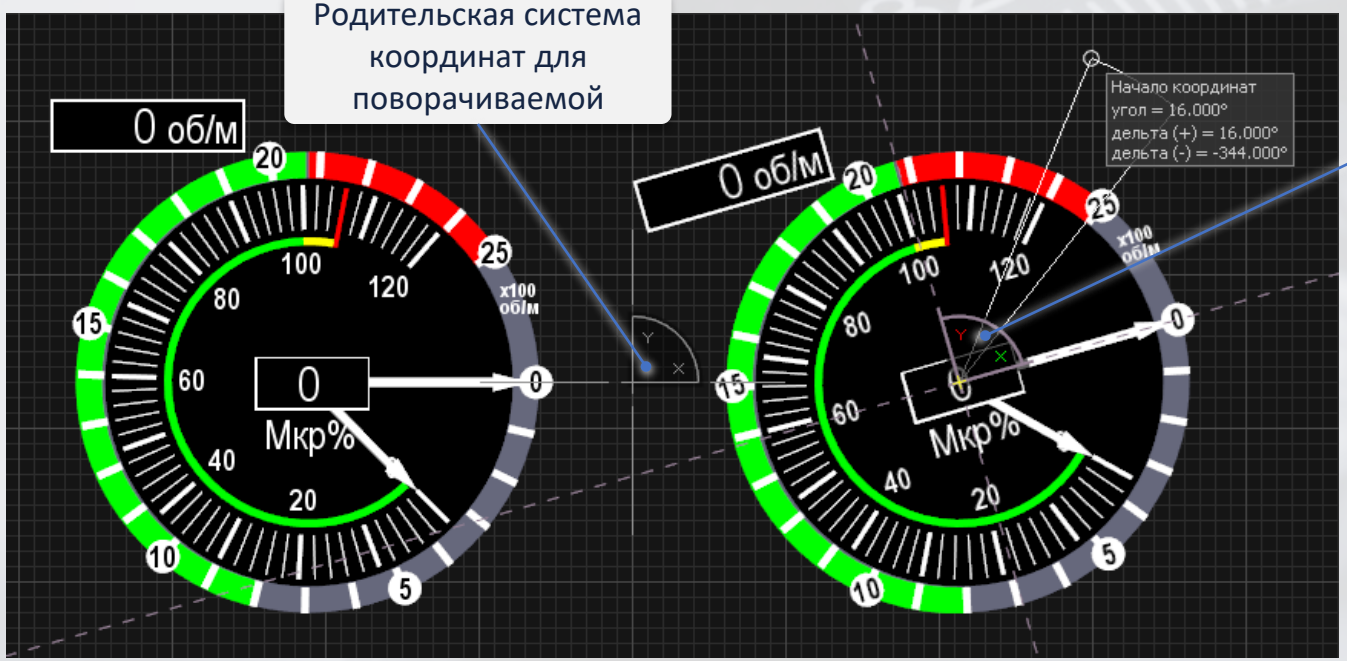
Параметры секущей линии и секущего прямоугольника при отработке проекта в динамике могут быть заданы формулами

Система координат

Примитивы в САПР строятся в некоторой системе координат. Если это не корневая система, то она называется локальной и располагается в приделах другой системы. В результате возникает иерархия вложенных друг в друга локальных систем координат, каждая из которых может быть смещена, повёрнута и масштабирована относительно родительской системы.

Это упрощает создание визуальных эффектов, а также компоновку графики на экране. Фрагменты индикационного кадра создаются в отдельных системах координат. Если необходимо сместить или повернуть фрагмент, то достаточно сместить или повернуть систему координат, в которой он создан, без изменения координат входящих в него примитивов.

На рисунке показан поворот системы координат правого индикатора. Все примитивы, построенные в системе координат, поворачиваются вместе с ней.

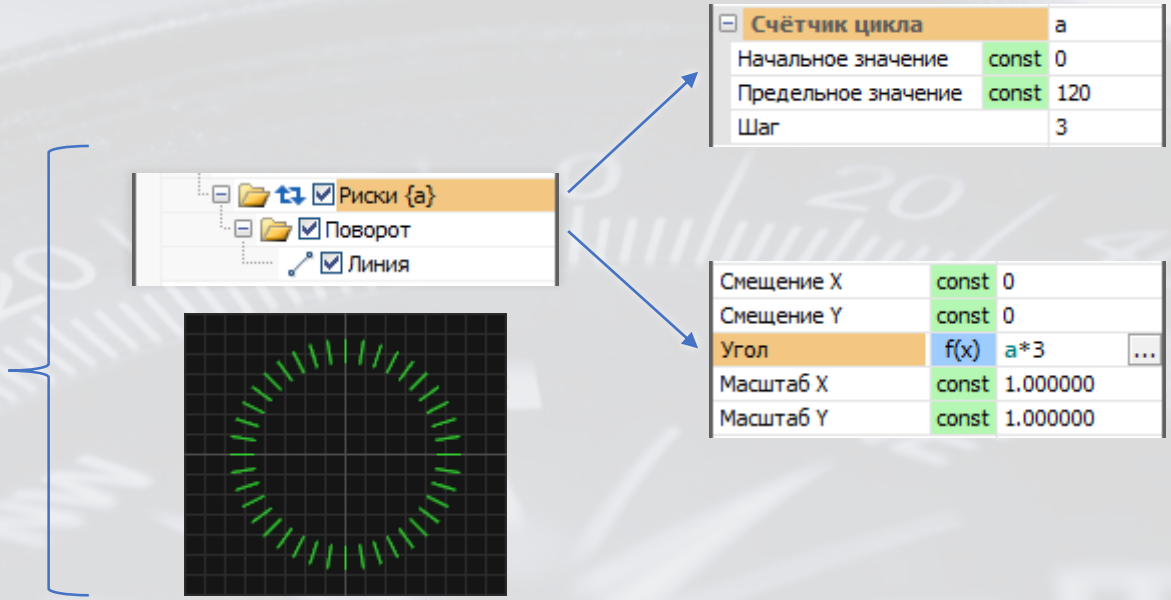


Выполнение в цикле

В программе есть возможность выполнить некоторую часть проекта заданное число раз, т.е. в цикле. Это позволяет упростить обработку некоторого массива данных или прорисовать повторяющиеся графические фрагменты, используя при этом меньшее число элементов проекта.

В примере задано циклическое выполнение содержимого группы Риски. Счётчиком цикла назначена переменная ‘a’. Её значение на каждом цикле будет меняться от 0 до 117 с шагом 3. К значению переменной ‘a’ привязан угол поворота системы координат дочерней группы Поворот. Она в свою очередь содержит единственный примитив Линия.

При отработке проекта группа Риски будет выполнена 40 раз с прорисовкой на каждом цикле одной линии под новым углом. Это даёт возможность всего 3 элементами проекта прорисовать 40 рисок, для отображения которых иначе потребовалось бы использовать 40 отдельных примитивов типа Линия.



Использование цикла также упрощает отображение промежуточных пунктов маршрута (ППМ). Они имеют одинаковую прорисовку, но их количество и положение может меняться.

ППМ также может быть оформлен в проекте в виде отдельной функции прорисовки, о чём будет рассказано далее.

Механизм функций в САПР позволяет упростить, сократить по времени и удешевить процесс разработки. Функции предназначены для вынесения некоторой функциональности из основной части проекта. Это может быть некий механизм обработки данных или графический фрагмент.

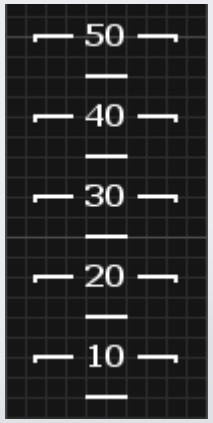
Разбивая проект на части и описывая их отдельно, мы упрощаем проект и делаем его более читабельным. Также это уменьшает число ошибок и позволяет использовать функции многократно как в данном, так и в других проектах.

Функции делятся на функции вычисления и функции прорисовки.

- **Функции вычисления**. Не могут выполнять прорисовку, но возвращают значения, что даёт возможность вызывать их из формул.
- **Функции прорисовки**. Не возвращают значений, но могут содержать графические примитивы.

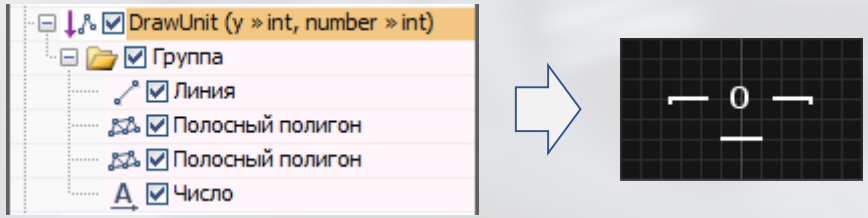
1

Давайте построим шкалу, показанную на рисунке ниже.



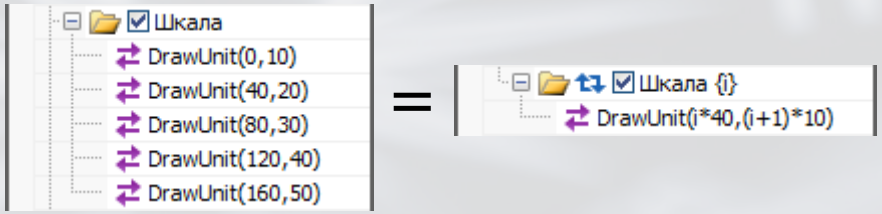
2

Она состоит из 5 повторяющихся элементов. Создадим для отображения такого элемента функцию прорисовки DrawUnit, включающую 4 графических примитива. В функцию при её вызове будем передавать позицию элемента по оси Y и номер, который необходимо отобразить.



3

Прорисовать шкалу при этом можно 5 отдельными вызовами функции DrawUnit. Либо разместив вызов функции в группе с циклическим выполнением. Счётчик цикла при этом будет изменяться от 0 до 4.

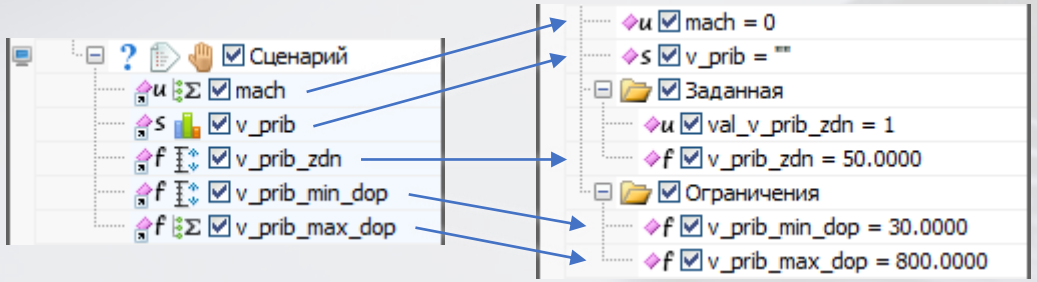


В первом случае число графических примитивов в проекте необходимых для построения шкалы сократилось в 1.8 раза, а во втором - в 2.5 раза.

Тестовые сценарии

Тестовые сценарии – это механизм подыгрыша САПР. Они эмулируют внешнюю информацию, поступающую в проект, и позволяют ‘оживить’ графику в нём. Также тестовые сценарии используются для проверки корректности работы фрагментов проекта, в том числе не связанных с отображением графики.

Тестовый сценарий включает набор элементов, каждый из которых ссылается на одноимённую входную переменную в проекте.

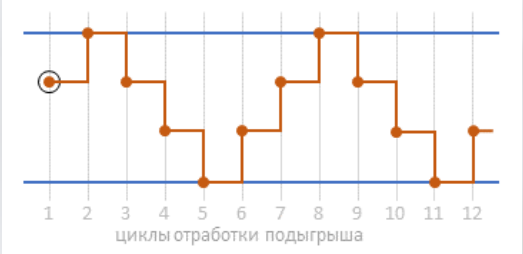


При отработке сценария, его элементы последовательно выполняются с присвоением новых значений входным переменным.

На каждый элемент сценария назначается один из 3 встроенных в САПР видов эмуляции данных.

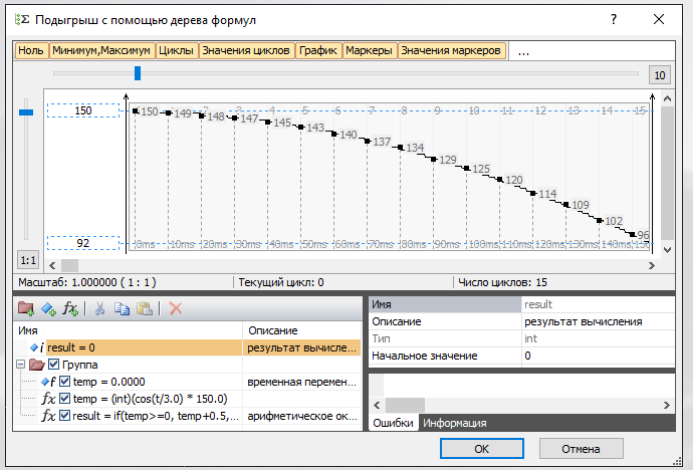
Линейный подыгрыш

Это повторяющееся изменение величины в заданных приделах с некоторым шагом.



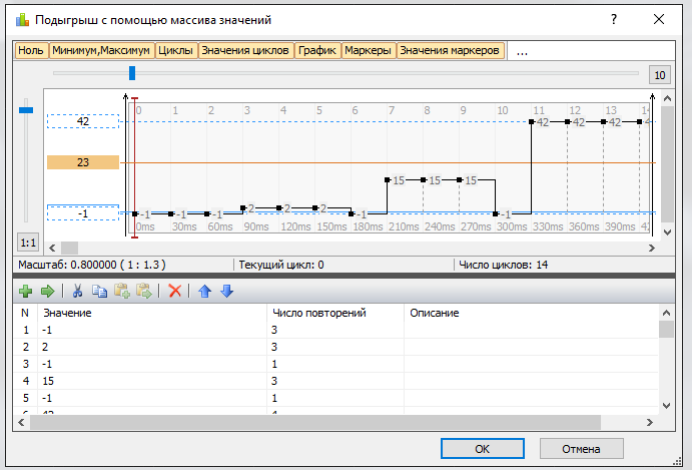
Подыгрыш с помощью дерева формул

Позволяет задать набор формул, результат вычисления которых будет являться значением для подыгрываемой входной переменной.



Подыгрыш с помощью массива значений

Содержит список значений, которые последовательно присваиваются входной переменной. После исчерпания списка происходит переход к первому значению.



Для подыгрыша входных переменных также могут быть использованы внешние подключаемые к САПР dll модули

Процесс моделирования

Программа имеет 2 режима работы:

- **Редактирование**. Режим, в котором САПР находится сразу после запуска. Предназначен для построения проекта и тестовых сценариев.
- **Моделирование**

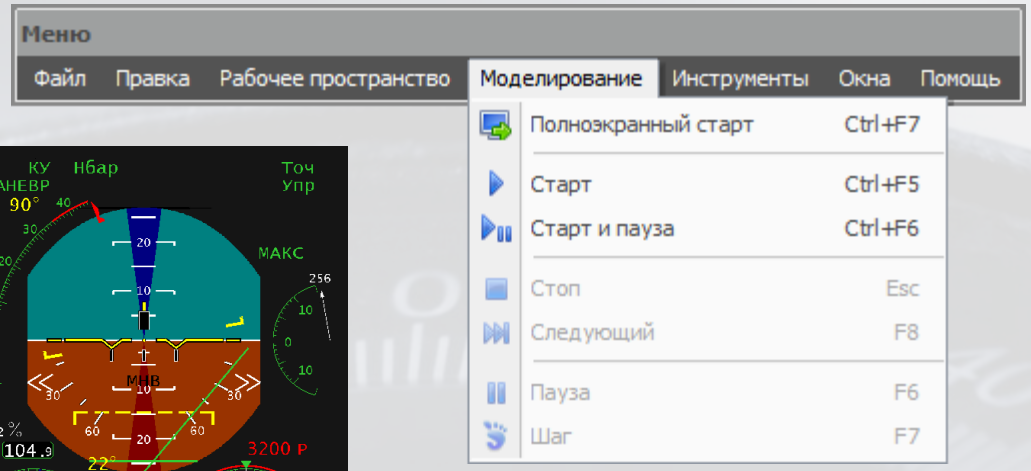
Моделирование – это циклический процесс, выполняемый с заданным временем синхронизации. На каждом его цикле происходит:

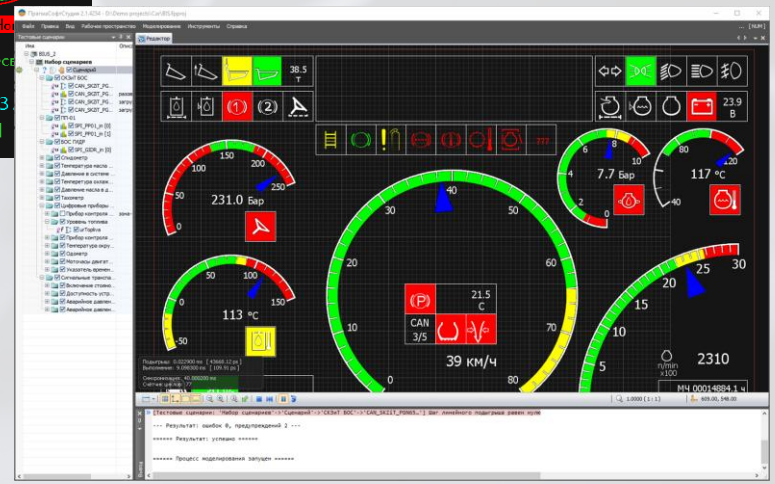
- Отработка тестовых сценариев с присвоением значений входным переменным.
- Обход дерева проекта. При этом вычисляются все необходимые формулы, вызываются функции вычисления и прорисовки, а также прорисовываются графические примитивы.
- Запись результатов отработки проекта в память (если нужно).

В свою очередь процесс моделирования делится на:

- **Оконный**. Графика прорисовывается в окне редактора. Пользователь имеет доступ к интерфейсу САПР, и может переключаться между тестовыми сценариями.
- **Полноэкранный**. Предназначен для финальной, презентационной, выставочной отработки проекта, так сказать, на показ. Либо, как основной режим работы при использовании в составе стендов.

Процесс моделирования может быть поставлен на паузу или пройден по шагам.





Генерация программного кода

САПР создаёт программный код на языке С, который соответствует требованиям критического с точки зрения безопасности встраиваемого программного обеспечения:

- Портируемость. Программный код не зависит от целевой аппаратно-программной среды и соответствует стандарту ANSI C (C99).
- Читаемость и структурированность на уровне функций и программных блоков.
- Отсутствие динамического выделения памяти.
- Отсутствие рекурсии, только ограниченные циклы.
- Ограниченное время выполнения.

Создаваемый программный код, может включать прорисовку с помощью одной из версий OpenGL: OpenGL 1.3, OpenGL 2.0, OpenGL ES 2.0 и OpenGL SC 1.0. Для версий OpenGL 1.3 и 2.0 программный код может быть сразу адоптирован для использования в операционной системе ОС Windows или ОС Linux.

Для всех вариантов генерации кода нами распространяются тестовые проекты для сред разработки: MS VisualStudio, QtCreator, AndroidStudio. Проекты могут быть использованы для изучения программного кода, либо, как заготовка для реального коммерческого проекта.

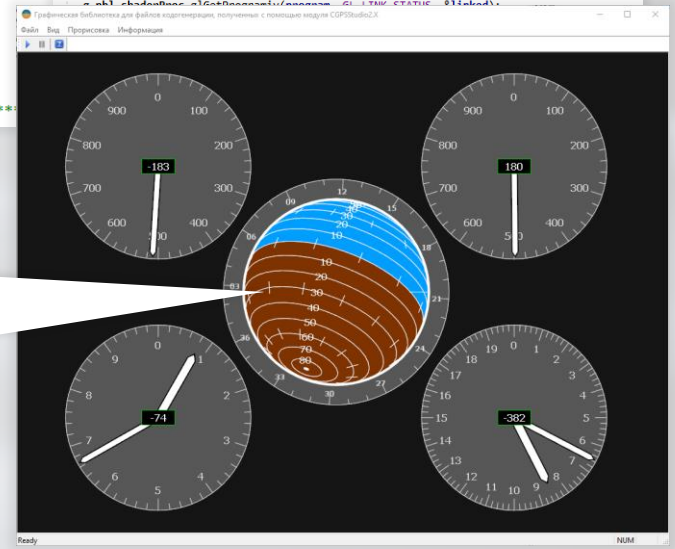
Программа позволяет смешивать программный код, который создаётся ей автоматически, с кодом, разработанным вручную, в том числе с прорисовкой с помощью библиотеки OpenGL.

```
/***** Группа ****************************************
* Путь: [Прорисовка: 'ОБРАМЛЕНИЕ']
*****************************************************/
/* Сохранение маскирования */
PssPushMask(10);
/* Режим маскирования */
PssSetMaskMode(PssMaskModeWithoutMask);

   /***** Группа ****************************************
   * Путь: [Прорисовка: 'ОБРАМЛЕНИЕ'->'Линии']
   *****************************************************/
   /* Сохранение маскирования */
   PssPushMask(1);
   /* Режим маскирования */
   PssSetMaskMode(PssMaskModeWithoutMask);

      /***** Примитив: линия ***************************
      * Путь: [Прорисовка: 'ОБРАМЛЕНИЕ'->'Линии'->'Линия']
      *****************************************************/
      /* Индекс цвета */
      PssSetColorIndex(13);
      /* Индекс толщины линии и диаметра точки */
      PssSetWidthIndex(1);
      /* Индекс типа линии */
      PssSetLineTypeIndex(0);
      /* Сглаживание */
      PssSetGraphMode(PssGraphModeSmooth, 1);
      /* Обводка */
      PssSetGraphMode(PssGraphModeEdging, 0);
      /* Прорисовка */
      PssDrawLine(-70,806, -70,-39);
      /*****************************************************
```

```
/**/
GLuint PBLLoadShader(GLenum shaderType, char const *source)
{  const GLuint shader = g_pbl_shaderProc.glCreateShader(shaderType);
   GLint compiled;
   if(shader==0)
      return 0;   /* error */
      /**/
   g_pbl_shaderProc.glShaderSource(shader, 1, &source, 0);
   g_pbl_shaderProc.glCompileShader(shader);
   compiled = GL_FALSE;
   g_pbl_shaderProc.glGetShaderiv(shader, GL_COMPILE_STATUS, &compiled);
   if(compiled!=GL_TRUE)
   {  g_pbl_shaderProc.glDeleteShader(shader);
      return 0;   /* error */
   }
   return shader;
}
/**/
GLuint PBLCreateProgram(char const *vertexShaderSource, char const *fragmentShaderSource)
{  const GLuint vertexShader = PBLLoadShader(GL_VERTEX_SHADER, vertexShaderSource);
   const GLuint pixelShader = PBLLoadShader(GL_FRAGMENT_SHADER, fragmentShaderSource);
   GLint linked;
      /**/
   const GLuint program = g_pbl_shaderProc.glCreateProgram();
   if(vertexShader==0 || pixelShader==0 || program==0)
   {  if(vertexShader!=0)
         g_pbl_shaderProc.glDeleteShader(vertexShader);
      if(pixelShader!=0)
         g_pbl_shaderProc.glDeleteShader(pixelShader);
      return 0;   /* error */
   }
      /**/
   g_pbl_shaderProc.glAttachShader(program, vertexShader);
   g_pbl_shaderProc.glDeleteShader(vertexShader);
   g_pbl_shaderProc.glAttachShader(program, pixelShader);
   g_pbl_shaderProc.glDeleteShader(pixelShader);
      /**/
   g_pbl_shaderProc.glLinkProgram(program);
   linked = GL_FALSE;
```



Программа, рисующая сферу в центре кадра, создана вручную с использованием OpenGL 2.0

Функциональность программы может быть увеличена без её изменения. Для этого используется механизм расширений в виде внешних подключаемых к САПР dll файлов. По желанию заказчика мы можем создать необходимые для него расширения, позволяющие, например, выполнять какую-либо специфическую проверку проекта или подыгрыша, добавлять удобную ему автоматизацию при работе с проектом, генерацию документации в необходимом формате и т.д..

Нами также оказывается поддержка как по электронной почте support@aviosoft.ru , так и на нашем форуме forum.aviosoft.ru , где Вы можете задать любой интересующий Вас вопрос о программе.

Также предлагаем Вам ознакомиться с online документацией на программу расположенную по адресу www.aviosoft.ru/psstudio/help .

Спасибо за внимание!

Наша почта: mail@aviosoft.ru