Интерфейс прикладного программирования геометрического ядра C3D

Его применение и главное отличие от API системы КОМПАС-3D

Аркадий Камнев

Рынок инженерного программного обеспечения представлен множеством универсальных систем автоматизированного проектирования, которые служат для повышения эффективности труда инженеров на предприятии. Системы трехмерного моделирования — одни из них. Они обладают мощной функциональностью твердотельного и поверхностного моделирования и широкими возможностями встроенных инструментальных средств. Но даже этих решений для выполнения конкретных производственных задач иногда бывает недостаточно.

Тогда на помощь специалистам приходит интерфейс прикладного программирования (API), то есть некоторый набор готовых классов, процедур, функций, структур и констант, призванный упростить создание пользовательских программных модулей, интегрированных с базовым программным продуктом. Наличие в CAD-системах API позволяет осуществлять тонкую настройку системы проектирования и может существенно сократить время разработки дополнительных программных компонентов к уже имеющейся на предприятии САПР. Так, базовая функциональность КОМПАС-3D легко расширяется за счет приложений для проектирования радиоэлектронной аппаратуры и электрооборудования, деталей машин, трубопроводов, металлоконструкций, зданий и сооружений различного назначения.

Однако, что делать, если ни одна из имеющихся систем автоматизированного проектирования не обладает должным спектром функциональных возможностей для ее полноценного использования специалистами? Выход есть — разработать собственную узкоспециализированную САПР!

Интерфейсы программирования приложений КОМПАС и C3D: что общего и в чем различие

API — это вспомогательный интерфейс разработчика программного обеспечения, позволяющий быстро создавать программы и компоненты к ним благодаря использованию готового набора функций, методов и процедур, представленных в максимально понятной и удобной для программиста форме. Как правило, внутренняя реализация таких объектов скрыта от пользователя и представляет собой предмет интеллектуальной собственности обладателя исходных кодов программного обеспечения.

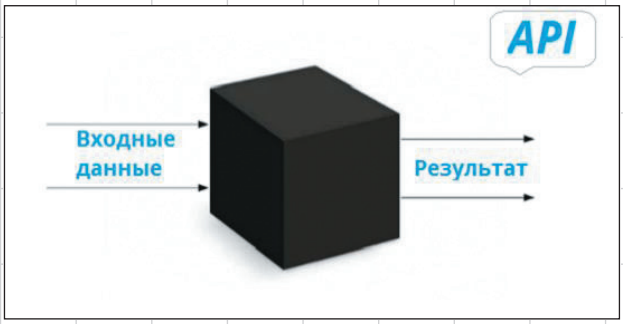

Рис. 1

Аркадий Камнев

Менеджер по продукту C3D в компании C3D Labs.

Особенности работы с интерфейсом прикладного программирования прописываются в технической документации. Здесь же указывается, какие значения необходимо подавать на вход конкретной функции, чтобы получить на ее выходе корректные значения, согласно предназначению данной функции. Таким образом, API позволяет абстрагироваться от реализации отдельных программных блоков при разработке приложений. Наиболее явно эта концепция раскрывается на примере черного ящика (рис. 1).

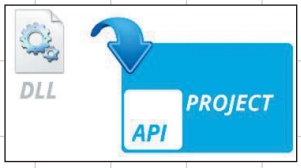

Рис. 2

Физически API-функции представляются в виде отдельного программного модуля, который динамически подключается извне к основному проекту в формате DLL-библиотеки (рис. 2).

Итак, что же такое «API геометрического ядра»?

Прежде всего — это набор всех экспортных методов ядра, которые использует в своей работе система автоматизированного проектирования, плюс внутренние объекты ядра в исходном виде: точки, матрицы, системы координат, кривые, поверхности, тела, вспомогательные объекты и др.

В то же время API КОМПАС является «оберткой» для избранных высокоуровневых методов, среди которых есть и методы ядра, и надстроенные над ними методы КОМПАС (в большей степени), позволяющие специалистам, разбирающимся в программировании, автоматизировать рутинные операции при работе с САПР. Это может быть новая панель, кнопка или отдельный внешний программный компонент, на который завязана качественно новая операция или, например, результирующая последовательного выполнения ряда функций, уже доступных в окружении КОМПАС.

Таким образом:

- API C3D предоставляет разработчику все низкоуровневые методы в исходном виде;
- API CAD — это надстройка над определенными высокоуровневыми методами CAD-системы.

Ниже приведен пример реализации на языке Python изменения цвета граней для простого макроса for _ in _ range (рис. 3):

Подключаем структуры и интерфейсы API:

```
import Kompas10API5 as KAPI
from win32com.client import Dispatch
import LDefin2D
import LDefin3D
```

Получаем головной интерфейс КОМПАС:

```
iKompasObject = Dispatch('KOMPAS.Application.5')
iKompasObject = KAPI.KompasObject(iKompasObject)
```

Получаем интерфейс 3D-документа:

```
iDocument3D = iKompasObject.ActiveDocument3D()
```

Берем у документа интерфейс менеджера селектированных объектов:

```
SlcMan = iDocument3D.GetSelectionMng()
```

Смотрим количество селектированных объектов для построения цикла обработки:

```
Count = SlcMan.GetCount()
```

Берем нулевой, начальный элемент:

```
n=0
Face0 = SlcMan.GetObjectByIndex (0)
```

Запоминаем у него цвет:

```
ColorFace0 = Face0.ColorParam()
color0 = ColorFace0.color
```

Затем в цикле для всех остальных селектированных объектов устанавливаем полученное значение цвета:

```
for n in range(0,Count,1):
 Face = SlcMan.GetObjectByIndex (n)
 ColorFace = Face.ColorParam()
 color = ColorFace.color
 ColorFace.color = color0
 Face.Update ()
SlcMan.UnselectAll ()
```

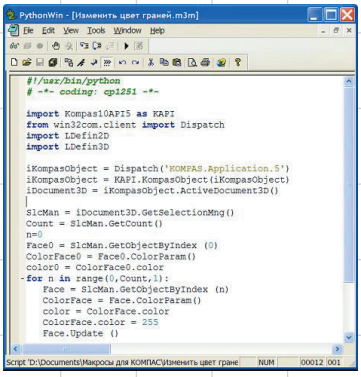

Рис. 3

Таким образом, получаем малую автоматизацию. Так, с помощью макроса мы можем выбрать сразу группу граней и окрасить их в цвет образцовой грани, тогда как, пользуясь стандартными средствами КОМПАС-3D, нам пришлось бы кликать в каждую грань, вызывать ее свойства, выбирать нужный цвет и применять заданные параметры.

Для наглядного сравнения двух интерфейсов приведем пример построения тела выдавливания на языке C++ средствами API C3D (рис. 4),

где `ExtrusionSolid` — метод, выполняющий построение тела выдавливания;

`sweptData` — информация об образующих кривых;

`direction` — направление выдавливания (вектор);

`solid1` — объект, до которого выполняется выдавливание в прямом направлении;

`solid2` — объект, до которого выполняется выдавливание в обратном направлении;

`checkIntersection` — флаг для объединения тел solid1 и solid2 с проверкой пересечения;

`params` — информация о способе выдавливания:

- `side1` — в прямом направлении;
- `side2` — в обратном направлении;
- `way==sw_scalarValue` — на длину scalarValue;
- `race` — с уклоном в соответствующую сторону;
- `thickness1` — с отступом наружу от образующей кривой;
- `thickness2` — с отступом внутрь от образующей кривой;
- `shellClosed` — с замкнутостью построенного тела;
- `checkSelfInt` — с проверкой результата построения на самопересечение;

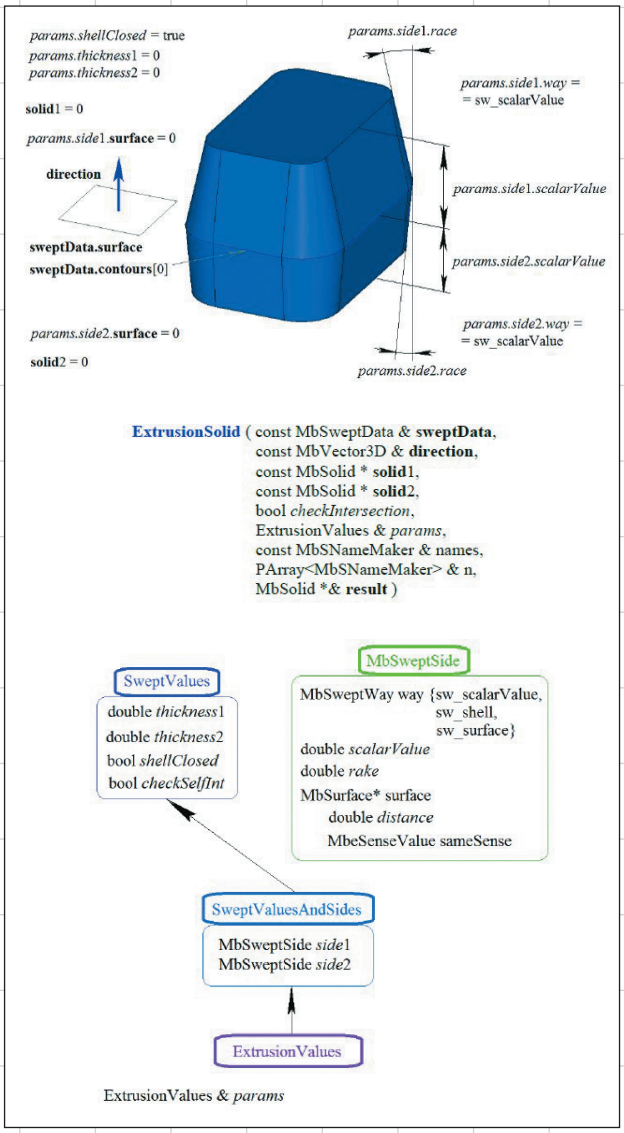

Рис. 4

`names` — именователь граней;

`n` — именователи сегментов образующих кривых;

`result` — построенная оболочка (тело).

То есть использование API C3D позволяет работать с базовыми объектами, лежащими в предметной области геометрического моделирования, и выполнять различные операции, а также преобразования над ними.

Инструкция по применению: API КОМПАС или API C3D?

Исходя из функциональных особенностей интерфейсов прикладного программирования системы КОМПАС-3D и лежащего в основе данной САПР геометрического ядра C3D, складывается набор признаков, по которым разработчики могут однозначно определить необходимость использования каждого из этих вспомогательных программных инструментов для успешного достижения поставленных перед ними целей и задач.

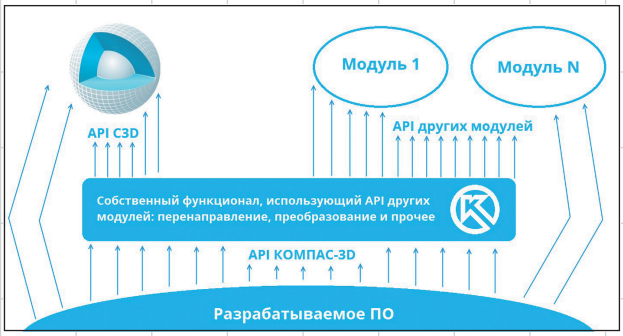

Рис. 5

Так, для разработки программ, расширяющих возможности стандартного комплекса КОМПАС-3D и являющихся неким плагином или дополнением к нему, достаточно использовать API КОМПАС. Причем никаких новых способностей в конечный продукт за счет API, будь то разработка необычных способов задания примитивов или построение нестандартных типов 2D- и 3D-геометрии, не может быть привлечено. Для функционирования такого программного обеспечения необходимо иметь предустановленную лицензионную версию системы КОМПАС-3D.

В случае если разработчику требуется в краткие сроки создать уникальный программный продукт без привязки к существующим САПР, необходимо использовать API C3D. Речь идет о создании специализированных CAD/CAM/CAE-систем или других типов инженерного ПО, функционирование которых связано с построением и обработкой геометрически точных 2D- и 3D-моделей.

На рис. 5 приведено наглядное изображение, иллюстрирующее, каким образом САПР КОМПАС-3D взаимодействует с различными API.

Пользователи C3D об использовании ядра в разработке их собственного программного обеспечения. Что влияет на выбор?

Александр Васильевич Щекин из Мордовского государственного университета им. Н.П. Огарева занимается разработкой модулей ЧПУ для системы трехмерного моделирования КОМПАС-3D:

- «Модуль ЧПУ. Токарная обработка» (рис. 6) предназначен для автоматизации разработки управляющих программ для токарных станков с ЧПУ (2-координатная обработка);
- «Модуль ЧПУ. Фрезерная обработка» служит для программирования фрезерных станков с ЧПУ (2,5-координатная обработка).

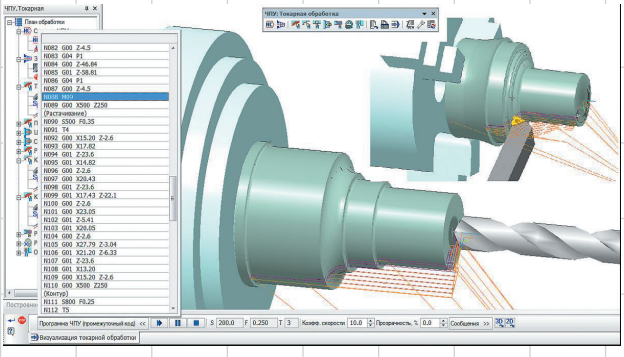

Рис. 6

Оба приложения работают с трехмерной моделью детали, созданной непосредственно в КОМПАС-3D, умеют автоматически рассчитывать траектории обработки, генерировать управляющую программу в G-коде и визуализировать обработку в окне системы. Вот как Александр комментирует использование API в процессе написания приложений:

Александр Щекин, Мордовский государственный университет им. Н.П. Огарева

«До появления на рынке программного обеспечения геометрического ядра C3D мы имели многолетний опыт разработки прикладных библиотек с применением API КОМПАС-3D. Однако для решения задач, в которых требовалось особое быстродействие при работе с геометрическими моделями, использование интерфейса программирования от КОМПАС оказалось не очень удобным. Особенности в производительности этого API объясняются тем, что работа с ним аналогична тому, как пользователь работает в среде КОМПАС-3D: чтобы выполнить какую-либо вычислительную операцию, сначала необходимо создать в документе геометрические объекты и только потом вызвать для них соответствующую математическую функцию API. Причем создание объекта в документе всегда связано с его записью во внутренние контейнеры КОМПАС, да и сами вызовы через COM-интерфейсы занимают некоторое время. Поэтому в тот период нам пришлось даже разработать набор собственных математических функций для 2D-вычислений, включая булевы операции с регионами.

К чему мы в конечном счете пришли?

В настоящее время мы используем смешанный подход на основе двух интерфейсов прикладного программирования. API КОМПАС по-прежнему применяется для разработки встроенного интерфейса, работы с файлами и для визуализации с помощью объектов внешней триангуляции в окне САПР КОМПАС-3D. API C3D мы используем в тех проектах, где требуется экстремальное быстродействие и экономия вычислительных ресурсов. Кстати, что касается визуализации через объекты внешней триангуляции, то API КОМПАС в этом отношении показывает высокую производительность и большее удобство, чем прямая работа с OpenGL».

Выбор, понятный каждому!

Итак, использование геометрического ядра C3D в процессе разработки программного обеспечения позволяет создавать принципиально новые кроссплатформенные CAD-, CAM- и CAE-системы без привязки к действующим САПР, проектировать мобильные и облачные приложения, а также дополнять функционал уже имеющихся программ возможностями 3D- и 2D-моделирования, задания параметрических ограничений и конвертации данных.

В то же время, когда необходимо дополнить и доработать систему трехмерного моделирования КОМПАС-3D, а также разработать новый программный модуль для действующих пользователей этой системы, достаточно использовать API КОМПАС-3D.

Узнайте больше о ядре геометрического моделирования C3D, его функциональных особенностях и условиях лицензирования на сайте *www.C3DLabs.com*. По всем вопросам обращайтесь в C3D Labs или в ближайший офис АСКОН.