# Плагины KiCad

**4 ноября 2021 г.**

# Содержание

*Система плагинов KiCad*

**Авторские права**



**Соавторы**

Cirilo Bernardo

**Перевод**

Барановский Константин <baranovskiykonstantin@gmail.com>, 2016-2019

**Обратная связь**

Оставить свои комментарии или замечания можно на следующих ресурсах:

- О документации KiCad: https://gitlab.com/kicad/services/kicad-doc/issues
- О программном обеспечении KiCad: https://gitlab.com/kicad/code/kicad/issues
- О переводе программного обеспечения KiCad: https://gitlab.com/kicad/code/kicad-i18n/issues

**Дата публикации**

29 января 2016 года

# 1 Введение в систему плагинов KiCad

Система плагинов KiCad - это специальный механизм для расширения возможностей KiCad, использующий динамические библиотеки. Одно из основный преимуществ использования плагинов — это отсутствие необходимости заново собирать весь проект KiCad в процессе разработки плагина. На деле, плагины можно построить с применением очень малого набора заголовочных файлов из всего дерева исходного кода KiCad. Освобождение от необходимости сборки KiCad в процессе разработки, здорово увеличивает продуктивность благодаря тому, что разработчик компилирует только тот код, который непосредственно относится к проектируемому плагину, что, в свою очередь, уменьшает время на каждую сборку в процессе тестирования.

Изначально, система плагинов была разработана для реализации предварительного просмотра 3D-моделей и обеспечения поддержки большего количества форматов 3D-моделей, без необходимости вносить серьезные изменения в исходный код KiCad для каждого нового поддерживаемого формата. Механизм плагинов со временем был обобщен и, таким образом, в будущем разработчики смогут создавать плагины различных классов. На данный момент в KiCad реализованы только 3D-плагины, но планируется добавить класс плагинов для печатных плат, который позволит пользователям реализовать импорт и экспорт данных.

## 1.1 Классы плагинов

Плагины делятся на классы, так как каждый из них решает проблемы определённой области и, поэтому, требует отдельного интерфейса к данной области. Например, плагины 3D-моделей загружают трёхмерные данные из файлов и преобразуют их в формат, который может быть показан в программе 3D-просмотра, в то время как плагин импорта/экспорта печатных плат должен принимать данные о печатных платах и экспортировать их в другой формат электрических или механических данных для KiCad. На данный момент разработан только класс 3D-плагинов и именно на нём будет сосредоточено внимание в этом документе.

Для реализации нового класса плагина необходимо добавить код в дерево исходного кода KiCad, который будет управлять загрузкой плагина. В файле plugins/ldr/pluginldr.h, из исходного кода KiCad, определён базовый класс для всех загрузчиков плагинов. В этом классе определены общие функции, которые должны присутствовать в любом из плагинов KiCad (шаблонный код), а их реализация будет выполнять основные проверки на совместимость версий между загрузчиком и доступными плагинами. Заголовочный файл plugins/ldr/3d/pluginldr3d.h определяет загрузчик для класса 3D-плагинов. Загрузчик отвечает за загрузку полученного плагина и делает его функции доступными для KiCad. Каждый экземпляр загрузчика плагинов предоставляет реализацию конкретного плагина и выступает в качестве прозрачного моста между kicad и функциями плагина. Для поддержки плагинов нужно не только добавить код загрузчика в исходный код KiCad, ещё нужен код для обнаружения плагинов и код для вызова функций плагина через загрузчик. В случае с 3D-плагином, обнаружение и вызов функций, вместе, реализовано в классе S3D_CACHE.

Разработчикам плагина не нужно разбираться в деталях исходного кода KiCad для управления им, если новый класс плагинов уже разработан. Для реализации плагина нужно лишь определить функции, объявленные в соответствующем классе плагинов.

Заголовочный файл include/plugins/kicad_plugin.h объявляет основные функции, обязательные для всех плагинов KiCad. Эти функции определяют имя класса плагина и имя данного плагина, возвращают информацию о версии API класса, информацию о версии самого плагина и проверяют их на совместимость. Вкратце об этих функциях:

```
/* b''Bb''b''ob''b''sb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''ив''b''мb''b' ↩
    'яb'' b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''лb''b''ab''b''гb''b''ив''b' ↩
    'нb''b''ab'' b''вb'' b''вb''b''ив''b''дb''b''eb'' b''cb''b''тb''b''pb''b''ob''b''кb''b' ↩
    'ив'' UTF-8 */
char const* GetKicadPluginClass( void );

/* b''Bb''b''ob''b''sb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''ив''b''нb''b' ↩
    'фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''ив''b''юb'' b''ob'' b''вb''b''eb''b''pb''b' ↩
    'cb''b''ив''b''ив'' API b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''лb''b''ab'' ↩
    b''гb''b''ив''b''нb''b''ab'' */
void GetClassVersion( unsigned char* Major, unsigned char* Minor,
     unsigned char* Patch, unsigned char* Revision );

/*
   b''Bb''b''ob''b''sb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''ив''b''cb''b' ↩
       'тb''b''ив''b''нb''b''yb'', b''eb''b''cb''b''лb''b''ив'' b''pb''b''eb''b''ab''b''лb'' ↩
       b''ив''b''sb''b''ob''b''вb''b''ab''b''нb''b''нb''b''ab''b''яb'' b''пb''b''pb''b''ob'' ↩
       b''вb''b''eb''b''pb''b''кb''b''ab'' b''вb''b''eb''b''pb''b''cb''b''ив''b''йb'' b' ↩
       'вb'' b''пb''b''лb''b''ab''b''гb''b''ив''b''нb''b''eb''
   b''ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''ив''b''лb''b''ab'', b''чb''b''тb''b' ↩
       'ob'' b''yb''b''кb''b''ab''b''sb''b''ab''b''нb''b''нb''b''ыb''b''йb'' API b''кb''b' ↩
       'лb''b''ab''b''cb''b''cb''b''ab'' -- b''cb''b''ob''b''вb''b''мb''b''eb''b''cb''b' ↩
       'тb''b''ив''b''мb''.
 */
bool CheckClassVersion( unsigned char Major,
    unsigned char Minor, unsigned char Patch, unsigned char Revision );

/* b''Bb''b''ob''b''sb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''ив''b''мb''b' ↩
    'яb'' b''дb''b''ab''b''нb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''лb''b''ab''b''гb''b' ↩
    'ив''b''нb''b''ab'', b''нb''b''ab''b''пb''b''pb''b''ив''b''мb''b''eb''b''pb'', " ↩
    PLUGIN_3D_VRML" */
const char* GetKicadPluginName( void );

/* b''Bb''b''ob''b''sb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''ив''b''нb''b' ↩
    'фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''ив''b''юb'' b''ob'' b''вb''b''eb''b''pb''b' ↩
    'cb''b''ив''b''ив'' b''дb''b''ab''b''нb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''лb''b' ↩
    'ab''b''гb''b''ив''b''нb''b''ab'' */
void GetPluginVersion( unsigned char* Major, unsigned char* Minor,
     unsigned char* Patch, unsigned char* Revision );
```

### 1.1.1 Класс плагинов PLUGIN_3D

В заголовочном файле include/plugins/3d/3d_plugin.h объявляются функции, которые должны быть реализованы в во всех 3D-плагинах, а также указано несколько функций, которые пользователь не должен изменять. Следующие функции не должны реализоваться пользователем:

```
/* b''Bb''b''ob''b''зb''b''вb''b''рb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''иb''b''мb''b' ↩
    'яb'' b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''лb''b''ab''b''гb''b''иb''b' ↩
    'нb''b''ab'' -- "PLUGIN_3D" */
char const* GetKicadPluginClass( void );

/* b''Bb''b''ob''b''зb''b''вb''b''рb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''иb''b''нb''b' ↩
    'фb''b''ob''b''рb''b''мb''b''ab''b''цb''b''иb''b''юb'' b''ob'' b''вb''b''eb''b''рb''b' ↩
    'cb''b''иb''b''иb'' API b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' PLUGIN_3D */
void GetClassVersion( unsigned char* Major, unsigned char* Minor,
     unsigned char* Patch, unsigned char* Revision );

/*
   b''Bb''b''ыb''b''пb''b''ob''b''лb''b''нb''b''яb''b''eb''b''тb'' b''ob''b''бb''b''ыb''b' ↩
       'чb''b''нb''b''уb''b''юb'' b''пb''b''рb''b''ob''b''вb''b''eb''b''рb''b''кb''b''уb'' b ↩
       ''вb''b''eb''b''рb''b''cb''b''иb''b''иb'', b''рb''b''eb''b''ab''b''лb''b''иb''b''зb'' ↩
       b''ob''b''вb''b''ab''b''нb''b''нb''b''уb''b''юb'' b''рb''b''ab''b''зb''b''рb''b''ab'' ↩
       b''бb''b''ob''b''тb''b''чb''b''иb''b''кb''b''ab''b''мb''b''иb''
   b''зb''b''ab''b''гb''b''рb''b''уb''b''зb''b''чb''b''иb''b''кb''b''ab'' b''кb''b''лb''b' ↩
       'ab''b''cb''b''cb''b''ab'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ob''b''вb'' ↩
       PLUGIN_3D, b''иb'' b''вb''b''ob''b''зb''b''вb''b''рb''b''ab''b''щb''b''ab''b''eb''b' ↩
       'тb'' b''иb''b''cb''b''тb''b''иb''b''нb''b''уb'', b''eb''b''cb''b''лb''b''иb''
   b''пb''b''рb''b''ob''b''вb''b''eb''b''рb''b''кb''b''ab'' b''уb''b''cb''b''пb''b''eb''b' ↩
       'шb''b''нb''b''ob'' b''пb''b''рb''b''ob''b''йb''b''дb''b''eb''b''нb''b''ab''
 */
bool CheckClassVersion( unsigned char Major, unsigned char Minor,
     unsigned char Patch, unsigned char Revision );
```

Следующие функции должны быть реализованы пользователем:

```
/*
   b''Bb''b''ob''b''зb''b''вb''b''рb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''кb''b''ob''b' ↩
       'лb''b''иb''b''чb''b''eb''b''cb''b''тb''b''вb''b''ob'' b''cb''b''тb''b''рb''b''ob''b' ↩
       'кb'' b''cb'' b''рb''b''ab''b''cb''b''шb''b''иb''b''рb''b''eb''b''нb''b''иb''b''яb''b ↩
       ''мb''b''иb'', b''кb''b''ob''b''тb''b''ob''b''рb''b''ыb''b''eb'' b''пb''b''ob''b' ↩
       'дb''b''дb''b''eb''b''рb''b''жb''b''иb''b''вb''b''ab''b''юb''b''тb''b''cb''b''яb''
   b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ob''b''мb''
 */
int GetNExtensions( void );

/*
   b''Bb''b''ob''b''зb''b''вb''b''рb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''зb''b''ab''b' ↩
       'пb''b''рb''b''ob''b''шb''b''eb''b''нb''b''нb''b''уb''b''юb'' b''cb''b''тb''b''рb''b' ↩
       'ob''b''кb''b''уb'' b''cb'' b''рb''b''ab''b''cb''b''шb''b''иb''b''рb''b''eb''b''нb''b ↩
       ''иb''b''eb''b''мb''; b''дb''b''ob''b''cb''b''тb''b''уb''b''пb''b''нb''b''ыb'' b' ↩
       'зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''яb'' b''ob''b''тb'' 0 b''дb''b' ↩
       'ob''
   GetNExtensions() - 1
 */
```

```
char const* GetModelExtension( int aIndex );

/*
   b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''ob''b''бb''b' ↩
       'щb''b''eb''b''eb'' b''кb''b''ob''b''лb''b''иb''b''чb''b''eb''b''cb''b''tb''b''вb''b' ↩
       'ob'' b''фb''b''иb''b''лb''b''ьb''b''tb''b''pb''b''ob''b''вb'' b''tb''b''иb''b''пb''b ↩
       ''ob''b''вb'' b''фb''b''ab''b''йb''b''лb''b''ob''b''вb'', b''кb''b''ob''b''tb''b' ↩
       'ob''b''pb''b''ыb''b''eb''
   b''пb''b''ob''b''дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''ab''b''юb''b''tb''b' ↩
       'cb''b''яb'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ob''b''мb''
 */
int GetNFilters( void );

/*
   b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''zb''b''ab''b' ↩
       'пb''b''pb''b''ob''b''шb''b''eb''b''нb''b''нb''b''ыb''b''йb'' b''фb''b''иb''b''лb''b' ↩
       'ьb''b''tb''b''pb'' b''tb''b''иb''b''пb''b''ob''b''вb'' b''фb''b''ab''b''йb''b''лb''b ↩
       ''ob''b''вb''; b''дb''b''ob''b''cb''b''tb''b''уb''b''пb''b''нb''b''ыb'' b''zb''b' ↩
       'нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''яb'' b''ob''b''tb'' 0 b''дb''b''ob''
   GetNFilters() - 1
 */
char const* GetFileFilter( int aIndex );

/*
    b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''иb''b''cb''b' ↩
        'tb''b''иb''b''нb''b''уb'', b''eb''b''cb''b''лb''b''иb'' b''пb''b''лb''b''ab''b' ↩
        'гb''b''иb''b''нb'' b''мb''b''ob''b''жb''b''eb''b''tb'' b''ob''b''tb''b''ob''b''бb'' ↩
        b''pb''b''ab''b''zb''b''иb''b''tb''b''ьb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b' ↩
        'йb'' b''tb''b''иb''b''пb'' 3D-b''мb''b''ob''b''дb''b''eb''b''лb''b''иb''.
    b''Bb'' b''нb''b''eb''b''кb''b''ob''b''tb''b''ob''b''pb''b''ыb''b''хb'' b''cb''b''лb''b ↩
        ''уb''b''чb''b''ab''b''яb''b''хb'', b''пb''b''лb''b''ab''b''гb''b''иb''b''нb'' b' ↩
        'нb''b''eb'' b''мb''b''ob''b''жb''b''eb''b''tb'' b''пb''b''pb''b''eb''b''дb''b''ob'' ↩
        b''cb''b''tb''b''ab''b''вb''b''иb''b''tb''b''ьb'' b''вb''b''иb''b''зb''b''уb''b' ↩
        'ab''b''лb''b''ьb''b''нb''b''уb''b''юb'' b''мb''b''ob''b''дb''b''eb''b''лb''b''ьb''
    b''иb'' b''дb''b''ob''b''лb''b''жb''b''eb''b''нb'' b''вb''b''eb''b''pb''b''нb''b''уb''b ↩
        ''tb''b''ьb'' b''лb''b''ob''b''жb''b''ьb''.
 */
bool CanRender( void );

/*
   b''3b''b''ab''b''гb''b''pb''b''уb''b''зb''b''иb''b''tb''b''ьb'' b''уb''b''кb''b''ab''b' ↩
       'зb''b''ab''b''нb''b''нb''b''уb''b''юb'' b''мb''b''ob''b''дb''b''eb''b''лb''b''ьb'' b ↩
       ''иb'' b''вb''b''eb''b''pb''b''нb''b''уb''b''tb''b''ьb'' b''уb''b''кb''b''ab''b''зb'' ↩
       b''ab''b''tb''b''eb''b''лb''b''ьb'' b''нb''b''ab'' b''дb''b''ab''b''нb''b''нb''b' ↩
       'ыb''b''eb'' b''eb''b''ёb'' b''вb''b''иb''b''зb''b''уb''b''ab''b''лb''b''ьb''b''нb''b ↩
       ''ob''b''гb''b''ob''
   b''пb''b''pb''b''eb''b''дb''b''cb''b''tb''b''ab''b''вb''b''лb''b''eb''b''нb''b''иb''b' ↩
       'яb''
```

```
 */
SCENEGRAPH* Load( char const* aFileName );
```

# 2 Примеры: Класс 3D-плагинов

Этот раздел содержит описание двух очень простых плагинов из класса PLUGIN_3D и проведёт пользователя от настройки до сборки кода.

## 2.1 Простой 3D-плагин

Этот пример проведёт пользователя через весь процесс разработки очень простого 3D-плагина под именем "PLUGIN_3D_DEMO1
Цель этого примера — показать конструкцию элементарного 3D-плагина, который не делает ничего, кроме предоставления некоторых фильтров типов файлов, что позволит пользователям KiCad отфильтровать файлы в процессе выбора 3D-моделей. Показанный здесь код, является необходимым минимумом для любого 3D-плагина и может быть использован как шаблон для создания более функциональных плагинов.

В процессе сборки демонстрационного проекта понадобится следующее:

- CMake
- Заголовочные файлы плагина KiCad
- Библиотека графа сцены KiCad (kicad_3dsg)

Для автоматического обнаружения заголовочных файлов KiCad и библиотеки нужно воспользоваться скриптом FindPackage на CMake. Скрипт, приведённый в этом примере, должен работать в Linux и MSWindows, если соответствующие заголовочные файлы установлены в `${KICAD_ROOT_DIR}/kicad`, а библиотека графа сцены — в `${KICAD_ROOT_DIR}/lib`.

Для начала создайте каталог для проекта и скрипт FindPackage:

```
mkdir demo && cd demo
export DEMO_ROOT=${PWD}
mkdir CMakeModules && cd CMakeModules
cat > FindKICAD.cmake << _EOF
find_path( KICAD_INCLUDE_DIR kicad/plugins/kicad_plugin.h
    PATHS ${KICAD_ROOT_DIR}/include $ENV{KICAD_ROOT_DIR}/include
    DOC "Kicad plugins header path."
    )

if( NOT ${KICAD_INCLUDE_DIR} STREQUAL "KICAD_INCLUDE_DIR-NOTFOUND" )

    # b''пb''b''ob''b''пb''b''ыb''b''тb''b''кb''b''ab'' b''иb''b''зb''b''вb''b''лb''b''eb'' ↩
        b''чb''b''ьb'' b''иb''b''нb''b''фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''иb''b' ↩
        'юb'' b''ob'' b''вb''b''eb''b''pb''b''cb''b''иb''b''иb'' b''иb''b''зb'' b''фb''b' ↩
        'ab''b''йb''b''лb''b''ab'' sg_version.h
    find_file( KICAD_SGVERSION sg_version.h
```

```
        PATHS ${KICAD_INCLUDE_DIR}
        PATH_SUFFIXES kicad/plugins/3dapi
        NO_DEFAULT_PATH )

    if( NOT ${KICAD_SGVERSION} STREQUAL "KICAD_SGVERSION-NOTFOUND" )

        # b''ив''b''зb''b''вb''b''лb''b''eb''b''чb''b''eb''b''нb''b''ив''b''eb'' b''cb''b' ↩
            'тb''b''pb''b''ob''b''кb''b''ив'' "#define KICADSG_VERSION*"
        file( STRINGS ${KICAD_SGVERSION} _version REGEX "^#define.*KICADSG_VERSION.*" )

        foreach( SVAR ${_version} )
            string( REGEX MATCH KICADSG_VERSION_[M,A,J,O,R,I,N,P,T,C,H,E,V,I,S]* _VARNAME $ ↩
                {SVAR} )
            string( REGEX MATCH [0-9]+ _VALUE ${SVAR} )

            if( NOT ${_VARNAME} STREQUAL "" AND NOT ${_VALUE} STREQUAL "" )
                set( _${_VARNAME} ${_VALUE} )
            endif()

        endforeach()

        # b''пb''b''pb''b''ив''b''вb''b''eb''b''cb''b''тb''b''ив'' b''нb''b''eb''b''уb''b' ↩
            'кb''b''ab''b''зb''b''ab''b''нb''b''нb''b''ыb''b''eb'' b''чb''b''ab''b''тb''b' ↩
            'cb''b''ив'' b''вb''b''eb''b''pb''b''cb''b''ив''b''ив'' b''кb'' b''нb''b''уb''b' ↩
            'лb''b''юb''
        if( NOT _KICADSG_VERSION_MAJOR )
            set( _KICADSG_VERSION_MAJOR 0 )
        endif()

        if( NOT _KICADSG_VERSION_MINOR )
            set( _KICADSG_VERSION_MINOR 0 )
        endif()

        if( NOT _KICADSG_VERSION_PATCH )
            set( _KICADSG_VERSION_PATCH 0 )
        endif()

        if( NOT _KICADSG_VERSION_REVISION )
            set( _KICADSG_VERSION_REVISION 0 )
        endif()

        set( KICAD_VERSION ${_KICADSG_VERSION_MAJOR}.${_KICADSG_VERSION_MINOR}.${ ↩
            _KICADSG_VERSION_PATCH}.${_KICADSG_VERSION_REVISION} )
        unset( KICAD_SGVERSION CACHE )

    endif()
endif()
```

```
find_library( KICAD_LIBRARY
    NAMES kicad_3dsg
    PATHS
        ${KICAD_ROOT_DIR}/lib $ENV{KICAD_ROOT_DIR}/lib
        ${KICAD_ROOT_DIR}/bin $ENV{KICAD_ROOT_DIR}/bin
    DOC "Kicad scenegraph library path."
    )

include( FindPackageHandleStandardArgs )
FIND_PACKAGE_HANDLE_STANDARD_ARGS( KICAD
    REQUIRED_VARS
        KICAD_INCLUDE_DIR
        KICAD_LIBRARY
        KICAD_VERSION
    VERSION_VAR KICAD_VERSION )


mark_as_advanced( KICAD_INCLUDE_DIR )
set( KICAD_VERSION_MAJOR ${_KICADSG_VERSION_MAJOR} CACHE INTERNAL "" )
set( KICAD_VERSION_MINOR ${_KICADSG_VERSION_MINOR} CACHE INTERNAL "" )
set( KICAD_VERSION_PATCH ${_KICADSG_VERSION_PATCH} CACHE INTERNAL "" )
set( KICAD_VERSION_TWEAK ${_KICADSG_VERSION_REVISION} CACHE INTERNAL "" )
_EOF
```

KiCad и его заголовочные файлы для плагина должны быть установлены. Если они установлены в пользовательский каталог или в `/opt` в Linux, или используется Windows, то нужно определить переменную среды `KICAD_ROOT_DIR`, которая будет указывать на каталог kicad, содержащий каталоги `include` и `lib`. Для OSX, показанный здесь скрипт FindPackage, возможно, придётся немного подкорректировать.

Для настройки и сборки кода примера будет использоваться CMake, создайте файл скрипта `CMakeLists.txt`:

```
cd ${DEMO_ROOT}
cat > CMakeLists.txt << _EOF
# b''yb''b''kb''b''ab''b''zb''b''ab''b''tb''b''ьb'' b''иb''b''mb''b''яb'' b''пb''b''pb''b' ↩
    'ob''b''eb''b''kb''b''тb''b''ab''
project( PLUGIN_DEMO )

# b''пb''b''pb''b''ob''b''вb''b''eb''b''pb''b''иb''b''тb''b''ьb'', b''yb''b''cb''b''тb''b' ↩
    'ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''ab'' b''лb''b''иb'' b''нb''b''yb''b' ↩
    'жb''b''нb''b''ab''b''яb'' b''вb''b''eb''b''pb''b''cb''b''иb''b''яb'' CMake b''cb''b' ↩
    'ob'' b''вb''b''cb''b''eb''b''mb''b''иb'' b''нb''b''yb''b''жb''b''нb''b''ыb''b''mb''b' ↩
    'иb'' b''cb''b''вb''b''ob''b''йb''b''cb''b''тb''b''вb''b''ab''b''mb''b''иb''
cmake_minimum_required( VERSION 2.8.12 FATAL_ERROR )

# b''yb''b''kb''b''ab''b''zb''b''ab''b''tb''b''ьb'' CMake b''mb''b''eb''b''cb''b''тb''b' ↩
    'ob'' b''дb''b''лb''b''яb'' b''пb''b''ob''b''иb''b''cb''b''kb''b''ab'' b''cb''b''kb''b' ↩
    'pb''b''иb''b''пb''b''тb''b''ab'' FindKICAD
set( CMAKE_MODULE_PATH ${PROJECT_SOURCE_DIR}/CMakeModules )
```

```
# b''пb''b''ob''b''пb''b''ыb''b''тb''b''кb''b''ab'' b''нb''b''ab''b''йb''b''тb''b''иb'' b' ↩
    'уb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''ыb''b' ↩
    'eb'' b''зb''b''ab''b''гb''b''ob''b''лb''b''ob''b''вb''b''ob''b''чb''b''нb''b''ыb''b' ↩
    'eb'' b''фb''b''ab''b''йb''b''лb''b''ыb'' b''иb'' b''бb''b''иb''b''бb''b''лb''b''иb''b' ↩
    'ob''b''тb''b''eb''b''кb''b''yb'' KiCad
# b''иb'' b''ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''иb''b''тb''b''ьb'' b''пb''b' ↩
    'eb''b''pb''b''eb''b''мb''b''eb''b''нb''b''нb''b''ыb''b''eb'':
#     KICAD_INCLUDE_DIR
#     KICAD_LIBRARY
find_package( KICAD 1.0 REQUIRED )

# b''дb''b''ob''b''бb''b''ab''b''вb''b''иb''b''тb''b''ьb'' b''кb''b''ab''b''тb''b''ab''b' ↩
    'лb''b''ob''b''гb'' b''зb''b''ab''b''гb''b''ob''b''лb''b''ob''b''вb''b''ob''b''чb''b' ↩
    'нb''b''ыb''b''xb'' b''фb''b''ab''b''йb''b''лb''b''ob''b''вb'' kicad b''кb'' b''пb''b' ↩
    'уb''b''тb''b''яb''b''мb'' b''пb''b''ob''b''иb''b''cb''b''кb''b''ab'' b''кb''b''ob''b' ↩
    'мb''b''пb''b''иb''b''лb''b''яb''b''тb''b''ob''b''pb''b''ab''
include_directories( ${KICAD_INCLUDE_DIR}/kicad )

# b''cb''b''ob''b''зb''b''дb''b''ab''b''тb''b''ьb'' b''пb''b''лb''b''ab''b''гb''b''иb''b' ↩
    'нb'' b''cb'' b''иb''b''мb''b''eb''b''нb''b''eb''b''мb'' s3d_plugin_demo1
add_library( s3d_plugin_demo1 MODULE
    src/s3d_plugin_demo1.cpp
    )

_EOF
```

Первый демонстрационный проект очень прост. Он состоит из единственного файла без каких-либо внешних зависимостей (помимо зависимостей компилятора). Начнём с создания каталога для исходного кода:

```
cd ${DEMO_ROOT}
mkdir src && cd src
export DEMO_SRC=${PWD}
```

Теперь создайте файл исходного кода для самого плагина:

**s3d_plugin_demo1.cpp**

```
#include <iostream>

// b''вb'' b''зb''b''ab''b''гb''b''ob''b''лb''b''ob''b''вb''b''ob''b''чb''b''нb''b''ob''b' ↩
    'мb'' b''фb''b''ab''b''йb''b''лb''b''eb'' 3d_plugin.h b''ob''b''бb''b''ъb''b''яb''b' ↩
    'вb''b''лb''b''eb''b''нb''b''ыb'' b''фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''иb'', b' ↩
    'ob''b''бb''b''яb''b''зb''b''ab''b''тb''b''eb''b''лb''b''ьb''b''нb''b''ыb''b''eb'' b' ↩
    'дb''b''лb''b''яb''
// 3D-b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ob''b''вb''
#include "plugins/3d/3d_plugin.h"

// b''yb''b''кb''b''ab''b''жb''b''иb''b''тb''b''eb'' b''иb''b''нb''b''фb''b''ob''b''pb''b' ↩
    'мb''b''ab''b''цb''b''иb''b''юb'' b''ob'' b''вb''b''eb''b''pb''b''cb''b''иb''b''иb'' b' ↩
```

```
    'дb''b''ab''b''нb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''лb''b''ab''b''гb''b''иb''b' ↩
    'нb''b''ab''; b''нb''b''eb'' b''пb''b''уb''b''тb''b''ab''b''йb''b''тb''b''eb'' b''эb''b' ↩
    'тb''b''ob'' b''cb''
// b''вb''b''eb''b''pb''b''cb''b''иb''b''eb''b''йb'' b''кb''b''лb''b''ab''b''cb''b''cb''b' ↩
    'ab'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ab'', b''кb''b''ob''b''тb''b''ob''b' ↩
    'pb''b''ab''b''яb'' b''уb''b''кb''b''ab''b''зb''b''ab''b''нb''b''ab'' b''вb'' 3d_plugin. ↩
    h
#define PLUGIN_3D_DEMO1_MAJOR 1
#define PLUGIN_3D_DEMO1_MINOR 0
#define PLUGIN_3D_DEMO1_PATCH 0
#define PLUGIN_3D_DEMO1_REVNO 0

// b''pb''b''eb''b''ab''b''лb''b''иb''b''зb''b''уb''b''йb''b''тb''b''eb'' b''фb''b''уb''b' ↩
    'нb''b''кb''b''цb''b''иb''b''юb'', b''кb''b''ob''b''тb''b''ob''b''pb''b''ab''b''яb'' b' ↩
    'пb''b''pb''b''eb''b''дb''b''ob''b''cb''b''тb''b''ab''b''вb''b''лb''b''яb''b''eb''b' ↩
    'тb'' b''пb''b''ob''b''лb''b''ьb''b''зb''b''ob''b''вb''b''ab''b''тb''b''eb''b''лb''b' ↩
    'яb''b''мb'' b''иb''b''мb''b''яb'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ab''
const char* GetKicadPluginName( void )
{
    return "PLUGIN_3D_DEMO1";
}

// b''pb''b''eb''b''ab''b''лb''b''иb''b''зb''b''уb''b''йb''b''тb''b''eb'' b''фb''b''уb''b' ↩
    'нb''b''кb''b''цb''b''иb''b''юb'', b''кb''b''ob''b''тb''b''ob''b''pb''b''ab''b''яb'' b' ↩
    'пb''b''pb''b''eb''b''дb''b''ob''b''cb''b''тb''b''ab''b''вb''b''лb''b''яb''b''eb''b' ↩
    'тb'' b''пb''b''ob''b''лb''b''ьb''b''зb''b''ob''b''вb''b''ab''b''тb''b''eb''b''лb''b' ↩
    'яb''b''мb'' b''вb''b''eb''b''pb''b''cb''b''иb''b''юb'' b''пb''b''лb''b''ab''b''гb''b' ↩
    'иb''b''нb''b''ab''
void GetPluginVersion( unsigned char* Major, unsigned char* Minor,
    unsigned char* Patch, unsigned char* Revision )
{
    if( Major )
        *Major = PLUGIN_3D_DEMO1_MAJOR;

    if( Minor )
        *Minor = PLUGIN_3D_DEMO1_MINOR;

    if( Patch )
        *Patch = PLUGIN_3D_DEMO1_PATCH;

    if( Revision )
        *Revision = PLUGIN_3D_DEMO1_REVNO;

    return;
}

// b''кb''b''ob''b''лb''b''иb''b''чb''b''eb''b''cb''b''тb''b''вb''b''ob'' b''пb''b''ob''b' ↩
    'дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''ab''b''eb''b''мb''b''ьb''b''xb'' b' ↩
```

```
    'рb''b''ab''b''cb''b''шb''b''иb''b''рb''b''eb''b''нb''b''иb''b''йb''; b''нb''b''ab'' b' ↩
    'cb''b''иb''b''cb''b''тb''b''eb''b''мb''b''ab''b''xb'' *NIX b''pb''b''ab''b''cb''b''шb'' ↩
    b''иb''b''pb''b''eb''b''нb''b''иb''b''яb''
// b''уb''b''кb''b''ab''b''зb''b''ыb''b''вb''b''ab''b''юb''b''тb''b''cb''b''яb'' b''дb''b' ↩
    'вb''b''ab''b''жb''b''дb''b''ыb'' - b''ob''b''дb''b''нb''b''ob'' b''вb'' b''нb''b''иb''b ↩
    ''жb''b''нb''b''eb''b''мb'' b''pb''b''eb''b''гb''b''иb''b''cb''b''тb''b''pb''b''eb'', b' ↩
    'вb''b''тb''b''ob''b''pb''b''ob''b''eb'' - b''вb'' b''вb''b''eb''b''pb''b''xb''b''нb''b' ↩
    'eb''b''мb''
#ifdef _WIN32
    #define NEXTS 7
#else
    #define NEXTS 14
#endif

// b''кb''b''ob''b''лb''b''иb''b''чb''b''eb''b''cb''b''тb''b''вb''b''ob'' b''пb''b''ob''b' ↩
    'дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''ab''b''eb''b''мb''b''ыb''b''xb'' b' ↩
    'фb''b''иb''b''лb''b''ьb''b''тb''b''pb''b''ob''b''вb'' b''тb''b''иb''b''пb''b''ob''b' ↩
    'вb'' b''фb''b''ab''b''йb''b''лb''b''ob''b''вb''
#define NFILS 5

// b''ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''иb''b''тb''b''eb'' b''cb''b''тb''b' ↩
    'pb''b''ob''b''кb''b''иb'' b''cb'' b''pb''b''ab''b''cb''b''шb''b''иb''b''pb''b''eb''b' ↩
    'нb''b''иb''b''яb''b''мb''b''иb'' b''иb'' b''фb''b''иb''b''лb''b''ьb''b''тb''b''pb''b' ↩
    'ab''b''мb''b''иb'', b''кb''b''ob''b''тb''b''ob''b''pb''b''ыb''b''eb'' b''пb''b''ob''b' ↩
    'дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''ab''b''eb''b''тb''
// b''дb''b''ab''b''нb''b''нb''b''ыb''b''йb'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''
static char ext0[] = "wrl";
static char ext1[] = "x3d";
static char ext2[] = "emn";
static char ext3[] = "iges";
static char ext4[] = "igs";
static char ext5[] = "stp";
static char ext6[] = "step";

#ifdef _WIN32
static char fil0[] = "VRML 1.0/2.0 (*.wrl)|*.wrl";
static char fil1[] = "X3D (*.x3d)|*.x3d";
static char fil2[] = "IDF 2.0/3.0 (*.emn)|*.emn";
static char fil3[] = "IGESv5.3 (*.igs;*.iges)|*.igs;*.iges";
static char fil4[] = "STEP (*.stp;*.step)|*.stp;*.step";
#else
static char ext7[] = "WRL";
static char ext8[] = "X3D";
static char ext9[] = "EMN";
static char ext10[] = "IGES";
static char ext11[] = "IGS";
static char ext12[] = "STP";
static char ext13[] = "STEP";
```

```
static char fil0[] = "VRML 1.0/2.0 (*.wrl;*.WRL)|*.wrl;*.WRL";
static char fil1[] = "X3D (*.x3d;*.X3D)|*.x3d;*.X3D";
static char fil2[] = "IDF 2.0/3.0 (*.emn;*.EMN)|*.emn;*.EMN";
static char fil3[] = "IGESv5.3 (*.igs;*.iges;*.IGS;*.IGES)|*.igs;*.iges;*.IGS;*.IGES";
static char fil4[] = "STEP (*.stp;*.step;*.STP;*.STEP)|*.stp;*.step;*.STP;*.STP";
#endif

// b''ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''иb''b''тb''b''eb'' b''cb''b''тb''b' ↩
    'pb''b''yb''b''кb''b''тb''b''yb''b''pb''b''yb'' b''дb''b''лb''b''яb'' b''yb''b''дb''b' ↩
    'ob''b''бb''b''нb''b''ob''b''гb''b''ob'' b''дb''b''ob''b''cb''b''тb''b''yb''b''пb''b' ↩
    'ab'' b''кb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''мb''
// b''вb'' b''вb''b''иb''b''дb''b''eb'' b''cb''b''пb''b''иb''b''cb''b''кb''b''ob''b''вb'' b ↩
    ''cb''b''тb''b''pb''b''ob''b''кb'' b''pb''b''ab''b''cb''b''шb''b''иb''b''pb''b''eb''b' ↩
    'нb''b''иb''b''йb'' b''иb'' b''фb''b''иb''b''лb''b''ьb''b''тb''b''pb''b''ob''b''вb''
static struct FILE_DATA
{
    char const* extensions[NEXTS];
    char const* filters[NFILS];

    FILE_DATA()
    {
        extensions[0] = ext0;
        extensions[1] = ext1;
        extensions[2] = ext2;
        extensions[3] = ext3;
        extensions[4] = ext4;
        extensions[5] = ext5;
        extensions[6] = ext6;
        filters[0] = fil0;
        filters[1] = fil1;
        filters[2] = fil2;
        filters[3] = fil3;
        filters[4] = fil4;

#ifndef _WIN32
        extensions[7] = ext7;
        extensions[8] = ext8;
        extensions[9] = ext9;
        extensions[10] = ext10;
        extensions[11] = ext11;
        extensions[12] = ext12;
        extensions[13] = ext13;
#endif
        return;
    }

} file_data;
```

```
// b''вb''b''оb''b''зb''b''вb''b''рb''b''аb''b''щb''b''аb''b''еb''b''тb'' b''кb''b''оb''b' ↩
    'лb''b''иb''b''чb''b''еb''b''сb''b''тb''b''вb''b''оb'' b''рb''b''аb''b''сb''b''шb''b' ↩
    'иb''b''рb''b''еb''b''нb''b''иb''b''йb'', b''пb''b''оb''b''дb''b''дb''b''еb''b''рb''b' ↩
    'жb''b''иb''b''вb''b''аb''b''еb''b''мb''b''ыb''b''хb'' b''эb''b''тb''b''иb''b''мb'' b' ↩
    'пb''b''лb''b''аb''b''гb''b''иb''b''нb''b''оb''b''мb''
int GetNExtensions( void )
{
    return NEXTS;
}

// b''вb''b''оb''b''зb''b''вb''b''рb''b''аb''b''щb''b''аb''b''еb''b''тb'' b''сb''b''тb''b' ↩
    'рb''b''оb''b''кb''b''уb'' b''рb''b''аb''b''сb''b''шb''b''иb''b''рb''b''еb''b''нb''b' ↩
    'иb''b''яb'' b''пb''b''оb'' b''уb''b''кb''b''аb''b''зb''b''аb''b''нb''b''нb''b''оb''b' ↩
    'мb''b''уb'' b''иb''b''нb''b''дb''b''еb''b''кb''b''сb''b''уb''
char const* GetModelExtension( int aIndex )
{
    if( aIndex < 0 || aIndex >= NEXTS )
        return NULL;

    return file_data.extensions[aIndex];
}

// b''вb''b''оb''b''зb''b''вb''b''рb''b''аb''b''щb''b''аb''b''еb''b''тb'' b''кb''b''оb''b' ↩
    'лb''b''иb''b''чb''b''еb''b''сb''b''тb''b''вb''b''оb'' b''сb''b''тb''b''рb''b''оb''b' ↩
    'кb'' b''фb''b''иb''b''лb''b''ьb''b''тb''b''рb''b''оb''b''вb'', b''пb''b''рb''b''еb''b' ↩
    'дb''b''оb''b''сb''b''тb''b''аb''b''вb''b''лb''b''яb''b''еb''b''мb''b''ыb''b''хb'' b' ↩
    'эb''b''тb''b''иb''b''мb'' b''пb''b''лb''b''аb''b''гb''b''иb''b''нb''b''оb''b''мb''
int GetNFilters( void )
{
    return NFILS;
}

// b''вb''b''оb''b''зb''b''вb''b''рb''b''аb''b''щb''b''аb''b''еb''b''тb'' b''сb''b''тb''b' ↩
    'рb''b''оb''b''кb''b''уb'' b''фb''b''иb''b''лb''b''ьb''b''тb''b''рb''b''аb'' b''пb''b' ↩
    'оb'' b''уb''b''кb''b''аb''b''зb''b''аb''b''нb''b''нb''b''оb''b''мb''b''уb'' b''иb''b' ↩
    'нb''b''дb''b''еb''b''кb''b''сb''b''уb''
char const* GetFileFilter( int aIndex )
{
    if( aIndex < 0 || aIndex >= NFILS )
        return NULL;

    return file_data.filters[aIndex];
}

// b''вb''b''оb''b''зb''b''вb''b''рb''b''аb''b''щb''b''аb''b''еb''b''тb'' b''лb''b''оb''b' ↩
    'жb''b''ьb'', b''еb''b''сb''b''лb''b''иb'' b''пb''b''лb''b''аb''b''гb''b''иb''b''нb'' b' ↩
```

```
    'нb''b''eb'' b''пb''b''ob''b''дb''b''гb''b''ob''b''тb''b''ob''b''вb''b''иb''b''лb'' b' ↩
    'дb''b''ab''b''нb''b''нb''b''ыb''b''eb'' b''вb''b''иb''b''зb''b''уb''b''ab''b''лb''b' ↩
    'иb''b''зb''b''ab''b''цb''b''иb''b''иb''
bool CanRender( void )
{
    return false;
}

// b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' NULL b''пb''b' ↩
    'ob''b''кb''b''ab'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb'' b''нb''b''eb'' b''пb''b' ↩
    'ob''b''дb''b''гb''b''ob''b''тb''b''ob''b''вb''b''иb''b''тb'' b''дb''b''ab''b''нb''b' ↩
    'нb''b''ыb''b''eb'' b''вb''b''иb''b''зb''b''уb''b''ab''b''лb''b''иb''b''зb''b''ab''b' ↩
    'цb''b''иb''b''иb''
SCENEGRAPH* Load( char const* aFileName )
{
    // b''эb''b''тb''b''ob''b''тb'' b''пb''b''pb''b''иb''b''мb''b''иb''b''тb''b''иb''b' ↩
        'вb''b''нb''b''ыb''b''йb'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb'' b''нb''b''eb'' ↩
         b''пb''b''ob''b''дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''ab''b''eb''b' ↩
        'тb'' b''pb''b''eb''b''нb''b''дb''b''eb''b''pb''b''иb''b''нb''b''гb'' b''нb''b''иb'' ↩
       b''кb''b''ab''b''кb''b''иb''b''хb'' b''мb''b''ob''b''дb''b''eb''b''лb''b''eb''b' ↩
        'йb''
    return NULL;
}
```

Данный файл исходного кода содержит минимальный набор всех необходимых элементов для реализации 3D-плагина. Этот плагин не производит никаких данных для рендеринга моделей, но может дополнить KiCad списком поддерживаемых расширений файлов моделей и фильтров типов файлов в диалоговом окне выбора 3D-моделей. К тому же, в KiCad строка расширения используется для выбора плагинов, с помощью которых можно загрузить выбранные модели. Например, если выбрано расширение `wrl`, то KiCad будет вызывать каждый плагин, который объявил о поддержке этого расширения, до тех пор, пока один из них не вернёт данные визуализации. Фильтры файлов, предоставленные каждым из плагинов, передаются в диалоговое окно выбора 3D-моделей, чтобы улучшить процесс поиска.

Для сборки плагина:

```
cd ${DEMO_ROOT}
# b''эb''b''кb''b''cb''b''пb''b''ob''b''pb''b''тb''b''иb''b''pb''b''yb''b''йb''b''тb''b' ↩
    'eb'' KICAD_ROOT_DIR, b''eb''b''cb''b''лb''b''иb'' b''пb''b''ob''b''нb''b''ab''b''дb''b' ↩
    'ob''b''бb''b''иb''b''тb''b''cb''b''яb''
mkdir build && cd build
cmake .. && make
```

Плагин будет построен, но не установлен. Можно скопировать его в катталог, в котором хранятся плагины, установленные вместе с kicad, если желаете, чтобы он был загружен.

## 2.2 Сложный 3D-плагин

Этот пример проведёт пользователя через весь процесс разработки 3D-плагина под именем ”PLUGIN_3D_DEMO2”. Цель этого примера — показать конструкцию элементарного графа сцены, который kicad сможет отобразить. Плагин должен

поддерживать тип файлов `txt`. Кроме этого, данный файл должен существовать, чтобы менеджер кэша смог запустить плагин. Содержимое файла не обрабатывается плагином, вместо этого, он просто создаёт граф сцены, содержащий пару тетраэдров. В данном примере предполагается, что первый пример был завершен и файлы скриптов CMakeLists.txt и FindKICAD.cmake были созданы.

Поместите новый файл исходного кода в тот же каталог, в котором находится файл исходного кода из предыдущего примера, и дальше будет дополнен уже имеющийся файл CMakeLists.txt, чтобы построить этот пример. Так как данный плагин будет создавать граф сцены для KiCad, нужно подключить библиотеку графов сцены из KiCad — `kicad_3dsg`. Эта библиотека предоставляет набор классов, которые можно использовать для построения объекта графа сцены. Объект графа сцены — это вспомогательный формат данных визуализации, который используется менеджером кэша трехмерных данных (3D Cache Manager). Все плагины, поддерживающие модель визуализации должны преобразовывать данные моделей в граф сцены с помощью библиотеки.

Первым делом нужно дополнить CMakeLists.txt для сборки примера проекта:

```
cd ${DEMO_ROOT}
cat >> CMakeLists.txt << _EOF
add_library( s3d_plugin_demo2 MODULE
    src/s3d_plugin_demo2.cpp
    )

target_link_libraries( s3d_plugin_demo2 ${KICAD_LIBRARY} )
_EOF
```

Теперь перейдите в каталог с исходным кодом и создайте новый файл:

```
cd ${DEMO_SRC}
```

**s3d_plugin_demo2.cpp**

```
#include <cmath>
// b''ob''b''бb''b''ъb''b''яb''b''вb''b''лb''b''eb''b''нb''b''иb''b''яb'' b''иb''b''зb'' b' ↩
    'кb''b''лb''b''ab''b''cb''b''cb''b''ab'' 3D-b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b' ↩
    'ob''b''вb''
#include "plugins/3d/3d_plugin.h"
// b''иb''b''нb''b''тb''b''eb''b''pb''b''фb''b''eb''b''йb''b''cb'' b''дb''b''лb''b''яb'' b' ↩
    'pb''b''ab''b''бb''b''ob''b''тb''b''ыb'' b''cb'' b''бb''b''иb''b''бb''b''лb''b''иb''b' ↩
    'ob''b''тb''b''eb''b''кb''b''ob''b''йb'' b''гb''b''pb''b''ab''b''фb''b''ab'' b''cb''b' ↩
    'цb''b''eb''b''нb''b''ыb'' b''иb''b''зb'' KiCad
#include "plugins/3dapi/ifsg_all.h"

// b''иb''b''нb''b''фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''иb''b''яb'' b''ob'' b''вb''b' ↩
    'eb''b''pb''b''cb''b''иb''b''иb'' b''дb''b''лb''b''яb'' b''дb''b''ab''b''нb''b''нb''b' ↩
    'ob''b''гb''b''ob'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ab''
#define PLUGIN_3D_DEMO2_MAJOR 1
#define PLUGIN_3D_DEMO2_MINOR 0
#define PLUGIN_3D_DEMO2_PATCH 0
#define PLUGIN_3D_DEMO2_REVNO 0
```

```
// b''пb''b''pb''b''eb''b''дb''b''ob''b''cb''b''тb''b''ab''b''вb''b''лb''b''яb''b''eb''b' ↩
    'тb'' b''иb''b''мb''b''яb'' b''дb''b''ab''b''нb''b''нb''b''ob''b''гb''b''ob'' b''пb''b' ↩
    'лb''b''ab''b''гb''b''иb''b''нb''b''ab''
const char* GetKicadPluginName( void )
{
    return "PLUGIN_3D_DEMO2";
}

// b''пb''b''pb''b''eb''b''дb''b''ob''b''cb''b''тb''b''ab''b''вb''b''лb''b''яb''b''eb''b' ↩
    'тb'' b''вb''b''eb''b''pb''b''cb''b''иb''b''юb'' b''дb''b''ab''b''нb''b''нb''b''ob''b' ↩
    'гb''b''ob'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ab''
void GetPluginVersion( unsigned char* Major, unsigned char* Minor,
    unsigned char* Patch, unsigned char* Revision )
{
    if( Major )
        *Major = PLUGIN_3D_DEMO2_MAJOR;

    if( Minor )
        *Minor = PLUGIN_3D_DEMO2_MINOR;

    if( Patch )
        *Patch = PLUGIN_3D_DEMO2_PATCH;

    if( Revision )
        *Revision = PLUGIN_3D_DEMO2_REVNO;

    return;
}


// b''кb''b''ob''b''лb''b''иb''b''чb''b''eb''b''cb''b''тb''b''вb''b''ob'' b''пb''b''ob''b' ↩
    'дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''ab''b''eb''b''мb''b''ыb''b''хb'' b' ↩
    'pb''b''ab''b''cb''b''шb''b''иb''b''pb''b''eb''b''нb''b''иb''b''йb''
#ifdef _WIN32
#define NEXTS 1
#else
#define NEXTS 2
#endif

// b''кb''b''ob''b''лb''b''иb''b''чb''b''eb''b''cb''b''тb''b''вb''b''ob'' b''пb''b''ob''b' ↩
    'дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''ab''b''eb''b''мb''b''ыb''b''хb'' b' ↩
    'фb''b''иb''b''лb''b''ьb''b''тb''b''pb''b''ob''b''вb''
#define NFILS 1

static char ext0[] = "txt";

#ifdef _WIN32
static char fil0[] = "demo (*.txt)|*.txt";
```

```
#else
static char ext1[] = "TXT";

static char fil0[] = "demo (*.txt;*.TXT)|*.txt;*.TXT";
#endif


static struct FILE_DATA
{
    char const* extensions[NEXTS];
    char const* filters[NFILS];

    FILE_DATA()
    {
        extensions[0] = ext0;
        filters[0] = fil0;

#ifndef _WIN32
        extensions[1] = ext1;
#endif
        return;
    }

} file_data;


int GetNExtensions( void )
{
    return NEXTS;
}


char const* GetModelExtension( int aIndex )
{
    if( aIndex < 0 || aIndex >= NEXTS )
        return NULL;

    return file_data.extensions[aIndex];
}


int GetNFilters( void )
{
    return NFILS;
}


char const* GetFileFilter( int aIndex )
```

```
{
    if( aIndex < 0 || aIndex >= NFILS )
        return NULL;

    return file_data.filters[aIndex];
}


// b''вb''b''eb''b''pb''b''нb''b''ёb''b''тb'' b''иb''b''cb''b''тb''b''иb''b''нb''b''уb'', b ↩
    ''кb''b''ob''b''гb''b''дb''b''ab'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb'' b''cb''b' ↩
    'мb''b''ob''b''жb''b''eb''b''тb'' b''пb''b''pb''b''eb''b''дb''b''ob''b''cb''b''тb''b' ↩
    'ab''b''вb''b''иb''b''тb''b''ьb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''eb'' b''вb''b' ↩
    'иb''b''зb''b''уb''b''ab''b''лb''b''иb''b''зb''b''ab''b''цb''b''иb''b''иb''
bool CanRender( void )
{
    return true;
}


// b''cb''b''ob''b''зb''b''дb''b''ab''b''нb''b''иb''b''eb'' b''дb''b''ab''b''нb''b''нb''b' ↩
    'ыb''b''хb'' b''вb''b''иb''b''зb''b''уb''b''ab''b''лb''b''иb''b''зb''b''ab''b''цb''b' ↩
    'иb''b''иb''
SCENEGRAPH* Load( char const* aFileName )
{
    // b''Дb''b''лb''b''яb'' b''эb''b''тb''b''ob''b''гb''b''ob'' b''пb''b''pb''b''иb''b' ↩
        'мb''b''eb''b''pb''b''ab'' b''бb''b''уb''b''дb''b''eb''b''тb'' b''cb''b''ob''b''зb'' ↩
        b''дb''b''ab''b''нb'' b''тb''b''eb''b''тb''b''pb''b''ab''b''эb''b''дb''b''pb'' (tx1) ↩
        , b''cb''b''ob''b''дb''b''eb''b''pb''b''жb''b''ab''b''щb''b''иb''b''йb''b''cb''b' ↩
        'яb'' b''вb'' b''гb''b''pb''b''ab''b''фb''b''eb''
    // b''cb''b''цb''b''eb''b''нb''b''ыb'' SCENEGRAPH (VRML Transform) b''иb'' b''cb''b' ↩
        'ob''b''cb''b''тb''b''ob''b''яb''b''щb''b''иb''b''йb'' b''иb''b''зb'' b''чb''b''eb'' ↩
        b''тb''b''ыb''b''pb''b''eb''b''хb'' b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb''b' ↩
        'ob''b''вb''
    // b''гb''b''pb''b''ab''b''нb''b''eb''b''йb'' SGSHAPE (VRML Shape) b''дb''b''лb''b' ↩
        'яb'' b''кb''b''ab''b''жb''b''дb''b''ob''b''йb'' b''иb''b''зb'' b''eb''b''гb''b' ↩
        'ob'' b''cb''b''тb''b''ob''b''pb''b''ob''b''нb''. b''Kb''b''ab''b''жb''b''дb''b' ↩
        'ob''b''йb'' b''гb''b''pb''b''ab''b''нb''b''иb''
    // b''пb''b''pb''b''иb''b''cb''b''вb''b''ab''b''иb''b''вb''b''ab''b''eb''b''тb''b''cb'' ↩
        b''яb'' b''цb''b''вb''b''eb''b''тb'' (SGAPPEARANCE) b''иb'' SGFACESET (VRML Geometry ↩
        ->indexedFaceSet).
    // b''Kb''b''ab''b''жb''b''дb''b''ыb''b''йb'' SGFACESET b''cb''b''вb''b''яb''b''зb''b' ↩
        'ыb''b''вb''b''ab''b''eb''b''тb''b''cb''b''яb'' b''cb''b''ob'' b''cb''b''пb''b''иb'' ↩
        b''cb''b''кb''b''ob''b''мb'' b''вb''b''eb''b''pb''b''шb''b''иb''b''нb'' (SGCOORDS), ↩
        b''cb''b''пb''b''иb''b''cb''b''кb''b''ob''b''мb''
    // b''вb''b''eb''b''кb''b''тb''b''ob''b''pb''b''ob''b''вb'' (SGNORMALS) b''иb'' b''иb'' ↩
        b''нb''b''дb''b''eb''b''кb''b''cb''b''ab''b''мb''b''иb'' b''кb''b''ob''b''ob''b' ↩
        'pb''b''дb''b''иb''b''нb''b''ab''b''тb'' (SGCOORDINDEX). b''Ob''b''дb''b''нb''b' ↩
        'ab'' b''гb''b''pb''b''ab''b''нb''b''ьb''
```

```
// b''ив''b''сb''b''пb''b''оb''b''лb''b''ьb''b''зb''b''уb''b''еb''b''тb''b''сb''b''яв'' ↩
    b''дв''b''лв''b''яв'' b''пв''b''рв''b''ев''b''дв''b''сb''b''тв''b''ав''b''вв''b' ↩
   'лв''b''ев''b''нв''b''ив''b''яв'' b''ов''b''дв''b''нв''b''оb''b''йв'' b''ив''b''зв'' ↩
    b''сb''b''тв''b''оb''b''рв''b''оb''b''нв'', b''тв''b''ав''b''кв'' b''чв''b''тв''b' ↩
   'оb'' b''мв''b''оb''b''жв''b''нв''b''оb''
// b''ив''b''сb''b''пв''b''оb''b''лв''b''ьb''b''зb''b''оb''b''вв''b''ав''b''тв''b''ьb'' ↩
    b''вв''b''ев''b''кв''b''тв''b''оb''b''рв''b''ыb'' b''вв''b''ев''b''рв''b''шb''b' ↩
   'ив''b''нв''-b''гв''b''рв''b''ав''b''нв''b''ев''b''йв'' (per-vertex-per-face normals ↩
   ).
//
// b''Эb''b''тв''b''оb''b''тв'' b''тв''b''ев''b''тв''b''рв''b''ав''b''эb''b''дв''b' ↩
   'рв'' b''яв''b''вв''b''лв''b''яв''b''ев''b''тв''b''сb''b''яв'' b''дв''b''оb''b''чв'' ↩
   b''ев''b''рв''b''нв''b''ив''b''мв'', b''пв''b''оb'' b''оb''b''тв''b''нв''b''оb''b' ↩
   'шb''b''ев''b''нв''b''ив''b''юв'' b''кв'' b''эb''b''лв''b''ев''b''мв''b''ев''b''нв'' ↩
   b''тв''b''уb'' b''вв''b''ев''b''рв''b''хв''b''нв''b''ев''b''гв''b''оb''
// b''уb''b''рв''b''оb''b''вв''b''нв''b''яв'' SCENEGRAPH (tx0), b''вв'' b''кв''b''оb''b ↩
   ''тв''b''оb''b''рв''b''оb''b''мв'' b''сb''b''оb''b''дв''b''ев''b''рв''b''жв''b''ив'' ↩
   b''тв''b''сb''b''яв'' b''вв''b''тв''b''оb''b''рв''b''оb''b''йв'' b''дв''b''оb''b' ↩
   'чв''b''ев''b''рв''b''нв''b''ив''b''йв''
// SCENEGRAPH (tx2), b''кв''b''оb''b''тв''b''оb''b''рв''b''ыb''b''йв'' b''вв'' b''сb''b ↩
   ''вв''b''оb''b''юв'' b''оb''b''чв''b''ев''b''рв''b''ев''b''дв''b''ьb'', b''яв''b' ↩
   'вв''b''лв''b''яв''b''ев''b''тв''b''сb''b''яв'' b''рв''b''ев''b''зв''b''уb''b''лв''b ↩
   ''ьb''b''тв''b''ав''b''тв''b''оb''b''мв''
// b''пв''b''рв''b''ев''b''оb''b''бв''b''рв''b''ав''b''зв''b''оb''b''вв''b''ав''b''нв'' ↩
   b''ив''b''яв'' b''тв''b''ев''b''тв''b''рв''b''ав''b''эв''b''дв''b''рв''b''ав'' tx1 ( ↩
   b''пв''b''оb''b''вв''b''оb''b''рв''b''оb''b''тв'' + b''сb''b''мв''b''ев''b''щb''b' ↩
   'ев''b''нв''b''ив''b''ев''). b''Эb''b''тв''b''ив''b''мв'' b''бв''b''уb''b''дв''b' ↩
   'ев''b''тв''
// b''пв''b''оb''b''кв''b''ав''b''зв''b''ав''b''нв''b''оb'' b''кв''b''ав''b''кв'' b' ↩
   'пв''b''оb''b''вв''b''тв''b''оb''b''рв''b''нв''b''оb'' b''ив''b''сb''b''пв''b''оb''b ↩
   ''лв''b''ьb''b''зв''b''оb''b''вв''b''ав''b''тв''b''ьb'' b''эв''b''лв''b''ев''b''мв'' ↩
   b''ев''b''нв''b''тв''b''ыb'' b''вв'' b''ив''b''ев''b''рв''b''ав''b''рв''b''хв''b' ↩
   'ив''b''ив'' b''гв''b''рв''b''ав''b''фв''b''ав'' b''сb''b''цв''b''ев''b''нв''b' ↩
   'ыb''.

// b''оb''b''бb''b''ъb''b''яв''b''вв''b''лв''b''ев''b''нв''b''ив''b''ев'' b''вв''b' ↩
   'ев''b''рв''b''шb''b''ив''b''нв'' b''тв''b''ев''b''тв''b''рв''b''ав''b''эb''b''дв''b ↩
   ''рв''b''ав''
// face 1: 0, 3, 1
// face 2: 0, 2, 3
// face 3: 1, 3, 2
// face 4: 0, 1, 2
double SQ2 = sqrt( 0.5 );
SGPOINT vert[4];
vert[0] = SGPOINT( 1.0, 0.0, -SQ2 );
vert[1] = SGPOINT( -1.0, 0.0, -SQ2 );
vert[2] = SGPOINT( 0.0, 1.0, SQ2 );
vert[3] = SGPOINT( 0.0, -1.0, SQ2 );
```

```
// b''cb''b''ob''b''зb''b''дb''b''ab''b''нb''b''иb''b''eb'' b''ob''b''бb''b''ъb''b' ↩
    'eb''b''кb''b''тb''b''ab'' b''пb''b''рb''b''eb''b''ob''b''бb''b''рb''b''ab''b''зb''b ↩
    ''ob''b''вb''b''ab''b''нb''b''иb''b''яb'' b''вb''b''eb''b''рb''b''хb''b''нb''b''eb'' ↩
    b''гb''b''ob'' b''уb''b''рb''b''ob''b''вb''b''нb''b''яb''; b''ob''b''нb'' b''бb''b' ↩
    'уb''b''дb''b''eb''b''тb'' b''cb''b''ob''b''дb''b''eb''b''рb''b''жb''b''ab''b''тb''b ↩
    ''ьb''
// b''вb''b''cb''b''eb'' b''ob''b''cb''b''тb''b''ab''b''лb''b''ьb''b''нb''b''ыb''b' ↩
    'eb'' b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb''b''ыb'' b''гb''b''рb''b''ab''b''фb'' ↩
    b''ob''b''вb'' b''cb''b''цb''b''eb''b''нb''b''ыb''; b''ob''b''бb''b''ъb''b''eb''b' ↩
    'кb''b''тb'' b''пb''b''рb''b''eb''b''ob''b''бb''b''рb''b''ab''b''зb''b''ob''b''вb''b ↩
    ''ab''b''нb''b''иb''b''яb'' b''мb''b''ob''b''жb''b''eb''b''тb''
// b''cb''b''ob''b''дb''b''eb''b''рb''b''жb''b''ab''b''тb''b''ьb'' b''дb''b''ob''b' ↩
    'чb''b''eb''b''рb''b''нb''b''иb''b''eb'' b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb''b ↩
    ''ыb'' b''пb''b''рb''b''eb''b''ob''b''бb''b''рb''b''ab''b''зb''b''ob''b''вb''b''ab'' ↩
    b''нb''b''иb''b''яb'' b''иb''b''лb''b''иb'' b''гb''b''рb''b''ab''b''нb''b''иb''
IFSG_TRANSFORM* tx0 = new IFSG_TRANSFORM( true );

// b''cb''b''ob''b''зb''b''дb''b''ab''b''тb''b''ьb'' b''ob''b''бb''b''ъb''b''eb''b' ↩
    'кb''b''тb'' b''пb''b''рb''b''eb''b''ob''b''бb''b''рb''b''ab''b''зb''b''ob''b''вb''b ↩
    ''ab''b''нb''b''иb''b''яb'', b''вb'' b''кb''b''ob''b''тb''b''ob''b''рb''b''ob''b' ↩
    'мb'' b''бb''b''уb''b''дb''b''уb''b''тb'' b''хb''b''рb''b''ab''b''нb''b''иb''b''тb'' ↩
    b''cb''b''яb'' b''гb''b''рb''b''ab''b''нb''b''иb''
IFSG_TRANSFORM* tx1 = new IFSG_TRANSFORM( tx0->GetRawPtr() );

// b''дb''b''ob''b''бb''b''ab''b''вb''b''иb''b''тb''b''ьb'' b''гb''b''рb''b''ab''b' ↩
    'нb''b''ьb'', b''кb''b''ob''b''тb''b''ob''b''рb''b''ab''b''яb'' b''бb''b''уb''b' ↩
    'дb''b''eb''b''тb'' b''cb''b''лb''b''уb''b''жb''b''иb''b''тb''b''ьb'' b''ob''b''дb'' ↩
    b''нb''b''ob''b''йb'' b''иb''b''зb'' b''cb''b''тb''b''ob''b''рb''b''ob''b''нb'' b' ↩
    'тb''b''eb''b''тb''b''рb''b''ab''b''эb''b''дb''b''рb''b''ab'';
// b''гb''b''рb''b''ab''b''нb''b''иb'' b''cb''b''ob''b''cb''b''тb''b''ob''b''яb''b' ↩
    'тb'' b''иb''b''зb'' b''нb''b''ab''b''бb''b''ob''b''рb''b''ab'' b''cb''b''тb''b' ↩
    'ob''b''рb''b''ob''b''нb'' b''иb'' b''ab''b''тb''b''рb''b''иb''b''бb''b''уb''b''тb'' ↩
    b''ob''b''вb'' b''вb''b''нb''b''eb''b''шb''b''нb''b''eb''b''гb''b''ob'' b''вb''b' ↩
    'иb''b''дb''b''ab'' (appearances)
IFSG_SHAPE* shape = new IFSG_SHAPE( *tx1 );

// b''дb''b''ob''b''бb''b''ab''b''вb''b''иb''b''тb''b''ьb'' b''нb''b''ab''b''бb''b' ↩
    'ob''b''рb'' b''cb''b''тb''b''ob''b''рb''b''ob''b''нb''; b''ob''b''нb'' b''cb''b' ↩
    'ob''b''cb''b''тb''b''ob''b''иb''b''тb'' b''иb''b''зb'' b''cb''b''пb''b''иb''b''cb'' ↩
    b''кb''b''ab'' b''кb''b''ob''b''ob''b''рb''b''дb''b''иb''b''нb''b''ab''b''тb'', b' ↩
    'иb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ob''b''вb''
// b''кb''b''ob''b''ob''b''рb''b''дb''b''иb''b''нb''b''ab''b''тb'', b''cb''b''пb''b' ↩
    'иb''b''cb''b''кb''b''ab'' b''вb''b''eb''b''рb''b''шb''b''иb''b''нb'', b''иb''b' ↩
    'нb''b''дb''b''eb''b''кb''b''cb''b''ob''b''вb'' b''вb''b''eb''b''рb''b''шb''b''иb''b ↩
    ''нb'' b''иb'' b''мb''b''ob''b''жb''b''eb''b''тb'' b''cb''b''ob''b''дb''b''eb''b' ↩
    'рb''b''жb''b''ab''b''тb''b''ьb'' b''cb''b''пb''b''иb''b''cb''b''ob''b''кb''
```

```
// b''цb''b''вb''b''eb''b''тb''b''ob''b''вb'' b''иb'' b''иb''b''хb'' b''иb''b''нb''b' ↩
    'дb''b''eb''b''кb''b''cb''b''ыb''.

IFSG_FACESET* face = new IFSG_FACESET( *shape );

IFSG_COORDS* cp = new IFSG_COORDS( *face );
cp->AddCoord( vert[0] );
cp->AddCoord( vert[3] );
cp->AddCoord( vert[1] );

// b''иb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ыb'' b''кb''b''ob''b''ob''b''pb''b' ↩
    'дb''b''иb''b''нb''b''ab''b''тb'' - b''пb''b''pb''b''иb''b''мb''b''eb''b''чb''b' ↩
    'ab''b''нb''b''иb''b''eb'': b''иb''b''cb''b''пb''b''ob''b''лb''b''ьb''b''зb''b''уb'' ↩
    b''юb''b''тb''b''cb''b''яb'' b''тb''b''pb''b''eb''b''уb''b''гb''b''ob''b''лb''b' ↩
    'ьb''b''нb''b''иb''b''кb''b''иb'';
// b''нb''b''ab'' b''пb''b''pb''b''ab''b''кb''b''тb''b''иb''b''кb''b''eb'', b''пb''b' ↩
    'лb''b''ab''b''гb''b''иb''b''нb''b''ыb'', b''кb''b''ob''b''тb''b''ob''b''pb''b''ыb'' ↩
    b''eb'' b''нb''b''eb'' b''мb''b''ob''b''гb''b''уb''b''тb'' b''ob''b''пb''b''pb''b' ↩
    'eb''b''дb''b''eb''b''лb''b''иb''b''тb''b''ьb'' b''кb''b''ab''b''кb''b''ab''b''яb''
// b''иb''b''зb'' b''cb''b''тb''b''ob''b''pb''b''ob''b''нb'' b''тb''b''pb''b''eb''b' ↩
    'уb''b''гb''b''ob''b''лb''b''ьb''b''нb''b''иb''b''кb''b''ab'' b''бb''b''уb''b''дb''b ↩
    ''eb''b''тb'' b''вb''b''иb''b''дb''b''нb''b''ab'', b''иb''b''cb''b''пb''b''ob''b' ↩
    'лb''b''ьb''b''зb''b''уb''b''юb''b''тb'' b''пb''b''ob'' b''дb''b''вb''b''eb'' b' ↩
    'тb''b''ob''b''чb''b''кb''b''иb''
// b''дb''b''лb''b''яb'' b''кb''b''ab''b''жb''b''дb''b''ob''b''гb''b''ob'' b''тb''b' ↩
    'pb''b''eb''b''уb''b''гb''b''ob''b''лb''b''ьb''b''нb''b''иb''b''кb''b''ab''
IFSG_COORDINDEX* coordIdx = new IFSG_COORDINDEX( *face );
coordIdx->AddIndex( 0 );
coordIdx->AddIndex( 1 );
coordIdx->AddIndex( 2 );

// b''ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''иb''b''тb''b''ьb'' b''ab''b' ↩
    'тb''b''pb''b''иb''b''бb''b''уb''b''тb''b''ыb''; b''ab''b''тb''b''pb''b''иb''b''бb'' ↩
    b''уb''b''тb''b''ыb'' b''пb''b''pb''b''иb''b''нb''b''ab''b''дb''b''лb''b''eb''b' ↩
    'жb''b''ab''b''тb'' b''гb''b''pb''b''ab''b''нb''b''иb''

// b''пb''b''уb''b''pb''b''пb''b''уb''b''pb''b''нb''b''ыb''b''йb''
IFSG_APPEARANCE* material = new IFSG_APPEARANCE( *shape);
material->SetSpecular( 0.1, 0.0, 0.1 );
material->SetDiffuse( 0.8, 0.0, 0.8 );
material->SetAmbient( 0.2, 0.2, 0.2 );
material->SetShininess( 0.2 );

// b''вb''b''eb''b''кb''b''тb''b''ob''b''pb''b''ыb''
IFSG_NORMALS* np = new IFSG_NORMALS( *face );
SGVECTOR nval = S3D::CalcTriNorm( vert[0], vert[3], vert[1] );
np->AddNormal( nval );
np->AddNormal( nval );
```

```
    np->AddNormal( nval );

    //
    // b''Гb''b''pb''b''ab''b''нb''b''ьb'' 2
    // b''Пb''b''pb''b''иb''b''мb''b''eb''b''чb''b''ab''b''нb''b''иb''b''eb'': b''ob''b' ↩
        'бb''b''ёb''b''pb''b''тb''b''кb''b''ab'' IFSG* b''иb''b''cb''b''пb''b''ob''b''лb''b' ↩
        'ьb''b''зb''b''уb''b''eb''b''тb''b''cb''b''яb'' b''пb''b''ob''b''вb''b''тb''b''ob''b ↩
        ''pb''b''нb''b''ob'' b''дb''b''лb''b''яb'' b''cb''b''ob''b''зb''b''дb''b''ab''b' ↩
        'нb''b''иb''b''яb'' b''иb''
    // b''уb''b''пb''b''pb''b''ab''b''вb''b''лb''b''eb''b''нb''b''иb''b''яb'' b''cb''b' ↩
        'тb''b''pb''b''уb''b''кb''b''тb''b''уb''b''pb''b''ab''b''мb''b''иb'' b''дb''b''ab''b ↩
        ''нb''b''нb''b''ыb''b''хb''.
    //
    shape->NewNode( *tx1 );
    face->NewNode( *shape );
    coordIdx->NewNode( *face );
    cp->NewNode( *face );
    np->NewNode( *face );

    // b''вb''b''eb''b''pb''b''шb''b''иb''b''нb''b''ыb''
    cp->AddCoord( vert[0] );
    cp->AddCoord( vert[2] );
    cp->AddCoord( vert[3] );

    // b''иb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ыb''
    coordIdx->AddIndex( 0 );
    coordIdx->AddIndex( 1 );
    coordIdx->AddIndex( 2 );

    // b''вb''b''eb''b''кb''b''тb''b''ob''b''pb''b''ыb''
    nval = S3D::CalcTriNorm( vert[0], vert[2], vert[3] );
    np->AddNormal( nval );
    np->AddNormal( nval );
    np->AddNormal( nval );
    // b''цb''b''вb''b''eb''b''тb'' (b''кb''b''pb''b''ab''b''cb''b''нb''b''ыb''b''йb'')
    material->NewNode( *shape );
    material->SetSpecular( 0.2, 0.0, 0.0 );
    material->SetDiffuse( 0.9, 0.0, 0.0 );
    material->SetAmbient( 0.2, 0.2, 0.2 );
    material->SetShininess( 0.1 );

    //
    // b''Гb''b''pb''b''ab''b''нb''b''ьb'' 3
    //
    shape->NewNode( *tx1 );
    face->NewNode( *shape );
    coordIdx->NewNode( *face );
    cp->NewNode( *face );
```

```
    np->NewNode( *face );

    // b''вb''b''eb''b''pb''b''шb''b''иb''b''нb''b''ыb''
    cp->AddCoord( vert[1] );
    cp->AddCoord( vert[3] );
    cp->AddCoord( vert[2] );

    // b''иb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ыb''
    coordIdx->AddIndex( 0 );
    coordIdx->AddIndex( 1 );
    coordIdx->AddIndex( 2 );

    // b''вb''b''eb''b''кb''b''тb''b''оb''b''pb''b''ыb''
    nval = S3D::CalcTriNorm( vert[1], vert[3], vert[2] );
    np->AddNormal( nval );
    np->AddNormal( nval );
    np->AddNormal( nval );

    // b''цb''b''вb''b''eb''b''тb'' (b''зb''b''eb''b''лb''b''ёb''b''нb''b''ыb''b''йb'')
    material->NewNode( *shape );
    material->SetSpecular( 0.0, 0.1, 0.0 );
    material->SetDiffuse( 0.0, 0.9, 0.0 );
    material->SetAmbient( 0.2, 0.2, 0.2 );
    material->SetShininess( 0.1 );

    //
    // b''Гb''b''pb''b''ab''b''нb''b''ьb'' 4
    //
    shape->NewNode( *tx1 );
    face->NewNode( *shape );
    coordIdx->NewNode( *face );
    cp->NewNode( *face );
    np->NewNode( *face );

    // b''вb''b''eb''b''pb''b''шb''b''иb''b''нb''b''ыb''
    cp->AddCoord( vert[0] );
    cp->AddCoord( vert[1] );
    cp->AddCoord( vert[2] );

    // b''иb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ыb''
    coordIdx->AddIndex( 0 );
    coordIdx->AddIndex( 1 );
    coordIdx->AddIndex( 2 );

    // b''вb''b''eb''b''кb''b''тb''b''оb''b''pb''b''ыb''
    nval = S3D::CalcTriNorm( vert[0], vert[1], vert[2] );
    np->AddNormal( nval );
    np->AddNormal( nval );
```

```
    np->AddNormal( nval );

    // b''цb''b''вb''b''eb''b''тb''  (b''cb''b''иb''b''нb''b''иb''b''йb'')
    material->NewNode( *shape );
    material->SetSpecular( 0.0, 0.0, 0.1 );
    material->SetDiffuse( 0.0, 0.0, 0.9 );
    material->SetAmbient( 0.2, 0.2, 0.2 );
    material->SetShininess( 0.1 );

    // b''cb''b''ob''b''зb''b''дb''b''ab''b''тb''b''ьb'' b''кb''b''ob''b''пb''b''иb''b' ↩
        'юb'' b''вb''b''cb''b''eb''b''гb''b''ob'' b''тb''b''eb''b''тb''b''pb''b''ab''b''эb'' ↩
        b''дb''b''pb''b''ab'', b''cb''b''мb''b''eb''b''cb''b''тb''b''иb''b''тb''b''ьb'' b' ↩
        'eb''b''ёb'' b''пb''b''ob'' b''ob''b''cb''b''иb'' Z b''нb''b''ab'' 2 (Z+2) b''иb''
    // b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b''ьb'' b''нb''b''ab'' 2/3PI
    IFSG_TRANSFORM* tx2 = new IFSG_TRANSFORM( tx0->GetRawPtr() );
    tx2->AddRefNode( *tx1 );
    tx2->SetTranslation( SGPOINT( 0, 0, 2 ) );
    tx2->SetRotation( SGVECTOR( 0, 0, 1 ), M_PI*2.0/3.0 );

    SGNODE* data = tx0->GetRawPtr();

    // b''yb''b''дb''b''ab''b''лb''b''иb''b''тb''b''ьb'' b''пb''b''eb''b''pb''b''eb''b' ↩
        'мb''b''eb''b''нb''b''нb''b''ыb''b''eb''
    delete shape;
    delete face;
    delete coordIdx;
    delete material;
    delete cp;
    delete np;
    delete tx0;
    delete tx1;
    delete tx2;

    return (SCENEGRAPH*)data;
}
```

# 3 Интерфейс программирования приложений (API)

Плагины создаются путём реализации интерфейса программирования приложений (Application Programming Interface — API). Каждый класс плагинов имеет свой уникальный API и в приведённых примерах 3D-плагинов была показана реализация API для класса 3D-плагинов, согласно объявлениям из заголовочного файла 3d_plugin.h. Кроме того, плагины могут использовать дополнительный API, объявленный в исходном коде KiCad. В случае с 3D-плагинами, все те плагины, что поддерживают визуализацию моделей, должны взаимодействовать используя API графов сцены, который объявлен в заголовочном файле ifsg_all.h и вложенных в него.

В этом разделе описываются детали API доступных классов плагинов и других API из KiCad, которые могут потребоваться

для реализации новых классов.

## 3.1 API класса плагинов

На данный момент доступен только один класс плагинов для KiCad — это класс 3D-плагинов. Все классы плагинов для KiCad должны реализовывать основной набор функций, объявленный в заголовочном файле kicad_plugin.h. Эти объявления можно рассматривать как базовый класс плагинов KiCad. Но на самом деле, реализации базового класса плагинов для KiCad не существует, эти заголовочные файлы присутствуют только для того, чтобы убедиться в том, что разработчики реализуют данные функции в каждом новом плагине.

В самом KiCad, каждый экземпляр загрузчика плагина реализует тот же API, что и плагин, так как этот загрузчик предоставляет все возможности данного класса. Это достигается тем, что класс загрузчика плагинов предоставляет открытый интерфейс, содержащий такие же имена функций, что и в реализации самого плагина. Список параметров может отличаться, чтобы можно было уведомить пользователя о возникновении каких-либо проблем, например, о том, что плагин не удалось загрузить. В процессе работы, загрузчик использует сохранённые указатели на каждую из функций API для их дальнейшего вызова по требованию пользователя.

### 3.1.1 API: базовый класс плагинов KiCad

Базовый класс плагинов KiCad определён в заголовочном файле kicad_plugin.h. Этот заголовочный файл должен подключаться ко всем другим классам плагинов. Для примера, посмотрите на объявления в заголовочном файле `3d_plugin.h` для класса 3D-плагинов. Прототипы этих функций кратко описаны в разделе Классы плагинов. В `pluginldr.cpp` показано как реализуется API базового загрузчика.

Чтобы понять назначение обязательных функций из заголовочного файла базового класса плагинов, нужно рассмотреть, что происходит при загрузке этих плагинов. В классе загрузчика объявляется виртуальная функция `Open()`, в которую передаётся полный путь к загружаемому плагину. В реализации функции `Open()` каждого конкретного класса загрузчика вызывается защищенная (protected) функция `open()` из базового загрузчика. Эта базовая функция `open()` пытается найти адреса каждой из обязательных функций базового плагина. Как только адреса для каждой из функций будут получены, начнётся выполнение следующих проверок:

1. Вызывается функция плагина `GetKicadPluginClass()` — возвращаемый результат сравнивается со значением из загрузчика для данного класса плагинов. Если значения не соответствуют, значит этот плагин не предназначен для работы с данным загрузчиком.

2. Вызывается функция плагина `GetClassVersion()` — возвращается версия API класса плагина, реализованная данным плагином.

3. Вызывается функция загрузчика `GetLoaderVersion()` — возвращается версия API класса плагина, реализованная данным загрузчиком.

4. В версиях API, полученных от плагина и загрузчика, должен совпадать главный номер версии (Major Version number), иначе считается что плагин с загрузчиком не совместимы. Это самая простая проверка на соответствие версий и выполняется она базовым загрузчиком плагина.

5. Вызывается функция плагина `CheckClassVersion()` — в функцию передаётся версия API класса плагинов, полученная от загрузчика. Если плагин поддерживает указанную версию — возвращается истина (`true`), подтверждая

совместимость. В таком случае загрузчик создаёт строку PluginInfo путём объединения результатов двух функций `GetKicadPluginName()` и `GetPluginVersion()`, и затем, процесс загрузки плагина продолжается с помощью функции `Open()` загрузчика.

### 3.1.2 API: класс 3D-плагинов

Класс 3D-плагинов объявлен в заголовочном файле `3d_plugin.h`. Помимо обязательных функций, в нем присутствуют дополнительные, их описание содержится в разделе Класс плагинов: PLUGIN_3D. Загрузчик для этого класса плагинов определён в pluginldr3d.cpp и помимо обязательных функций API, реализует следующие дополнительные общедоступные функции:

```
/* b''Ob''b''тb''b''кb''b''pb''b''ыb''b''тb''b''ьb'' b''пb''b''лb''b''ab''b''гb''b''иb''b' ↩
    'нb'', b''yb''b''кb''b''ab''b''зb''b''ab''b''нb''b''нb''b''ыb''b''йb'' b''вb'' b''вb''b' ↩
    'иb''b''дb''b''eb'' b''пb''b''ob''b''лb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''yb''b' ↩
    'тb''b''иb'' "aFullFileName" */
bool Open( const wxString& aFullFileName );

/* b''3b''b''ab''b''кb''b''pb''b''ыb''b''тb''b''ьb'', b''ob''b''тb''b''кb''b''pb''b''ыb''b' ↩
    'тb''b''ыb''b''йb'' b''вb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''йb'' b''мb''b''ob''b' ↩
    'мb''b''eb''b''нb''b''тb'', b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''.*/
void Close( void );

/* b''Пb''b''ob''b''лb''b''yb''b''чb''b''иb''b''тb''b''ьb'' b''вb''b''eb''b''pb''b''cb''b' ↩
    'иb''b''юb'' API b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''лb''b''ab''b''гb'' ↩
    b''иb''b''нb''b''ob''b''вb'', b''pb''b''eb''b''ab''b''лb''b''иb''b''зb''b''ob''b''вb''b' ↩
    'ab''b''нb''b''нb''b''yb''b''юb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''мb'' b''зb''b' ↩
    'ab''b''гb''b''pb''b''yb''b''зb''b''чb''b''иb''b''кb''b''ob''b''мb'' */
void GetLoaderVersion( unsigned char* Major, unsigned char* Minor,
    unsigned char* Revision, unsigned char* Patch ) const;
```

Необходимые функция из класса 3D-плагинов выявляются с помощью следующих функций:

```
/* b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''иb''b''мb''b' ↩
    'яb'' b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''лb''b''ab''b''гb''b''иb''b' ↩
    'нb''b''ob''b''вb'' b''иb''b''лb''b''иb'' NULL, b''eb''b''cb''b''лb''b''иb'' b''пb''b' ↩
    'лb''b''ab''b''гb''b''иb''b''нb'' b''нb''b''eb'' b''зb''b''ab''b''гb''b''pb''b''yb''b' ↩
    'жb''b''eb''b''нb'' */
char const* GetKicadPluginClass( void );

/* b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''лb''b''ob''b' ↩
    'жb''b''ьb'', b''eb''b''cb''b''лb''b''иb'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb'' b' ↩
    'нb''b''eb'' b''зb''b''ab''b''гb''b''pb''b''yb''b''жb''b''eb''b''нb'' */
bool GetClassVersion( unsigned char* Major, unsigned char* Minor,
    unsigned char* Patch, unsigned char* Revision );

/* b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''лb''b''ob''b' ↩
    'жb''b''ьb'', b''eb''b''cb''b''лb''b''иb'' b''вb''b''eb''b''pb''b''cb''b''иb''b''яb'' b' ↩
    'кb''b''лb''b''ab''b''cb''b''cb''b''ab'' b''нb''b''eb'' b''cb''b''ob''b''вb''b''мb''b' ↩
```

```
    'eb''b''cb''b''tb''b''ib''b''mb''b''ab'' b''ib''b''lb''b''ib'' b''pb''b''lb''b''ab''b' ↩
    'rb''b''ib''b''nb'' b''nb''b''eb'' b''zb''b''ab''b''rb''b''pb''b''yb''b''zb''b''eb''b' ↩
    'nb'' */
bool CheckClassVersion( unsigned char Major, unsigned char Minor,
    unsigned char Patch, unsigned char Revision );

/* b''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''wb''b''ab''b''eb''b''tb'' b''ib''b''mb''b' ↩
    'yb'' b''pb''b''lb''b''ab''b''rb''b''ib''b''nb''b''ab'' b''ib''b''lb''b''ib'' NULL, b' ↩
    'eb''b''cb''b''lb''b''ib'' b''pb''b''lb''b''ab''b''rb''b''ib''b''nb'' b''nb''b''eb'' b' ↩
    'zb''b''ab''b''rb''b''pb''b''yb''b''zb''b''eb''b''nb'' */
const char* GetKicadPluginName( void );

/*
   b''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''wb''b''ab''b''eb''b''tb'' b''lb''b''ob''b' ↩
       'zb''b''ab''b''rb''b''pb''b''yb''b''zb''b''eb''b''nb'', b''eb''b''cb''b''lb''b''ib'' b''pb''b''lb''b''ab''b''rb''b''ib''b''nb'' ↩
        b''nb''b''eb'' b''zb''b''ab''b''rb''b''pb''b''yb''b''zb''b''eb''b''nb'', b''vb'' b' ↩
       'pb''b''pb''b''ob''b''tb''b''ib''b''vb''b''nb''b''ob''b''mb'' b''cb''b''lb''b''yb''b' ↩
       'cb''b''ab''b''eb''
   b''vb'' b''pb''b''ab''b''pb''b''ab''b''mb''b''eb''b''tb''b''pb''b''ab''b''xb'' b''6b''b' ↩
       'yb''b''db''b''eb''b''tb'' b''cb''b''ob''b''db''b''eb''b''pb''b''zb''b''ab''b''tb''b' ↩
       'ib''b''cb''b''yb'' b''pb''b''eb''b''zb''b''yb''b''lb''b''ib''b''tb''b''ab''b''tb'' b ↩
       ''fb''b''yb''b''nb''b''kb''b''cb''b''ib''b''ib'' GetPluginInfo()
 */
bool GetVersion( unsigned char* Major, unsigned char* Minor,
    unsigned char* Patch, unsigned char* Revision );

/*
   b''eb''b''cb''b''lb''b''ib'' b''pb''b''lb''b''ab''b''rb''b''ib''b''nb'' b''nb''b''eb'' b ↩
       ''zb''b''ab''b''rb''b''pb''b''yb''b''zb''b''eb''b''nb'', b''yb''b''cb''b''tb''b''ab'' ↩
       b''nb''b''ab''b''vb''b''lb''b''ib''b''vb''b''ab''b''eb''b''tb'' b''zb''b''nb''b''ab'' ↩
       b''cb''b''eb''b''nb''b''ib''b''eb'' b''pb''b''eb''b''pb''b''eb''b''mb''b''eb''b''nb'' ↩
       b''nb''b''ob''b''йb'' aPluginInfo
   b''vb'' b''vb''b''ib''b''db''b''eb'' b''pb''b''yb''b''cb''b''tb''b''ob''b''йb'' b''cb''b ↩
       ''tb''b''pb''b''ob''b''kb''b''ib''; b''vb'' b''pb''b''pb''b''ob''b''tb''b''ib''b' ↩
       'vb''b''nb''b''ob''b''mb'' b''cb''b''lb''b''yb''b''cb''b''ab''b''eb'', b''zb''b''nb'' ↩
       b''ab''b''cb''b''eb''b''nb''b''ib''b''eb'' b''ob''b''6b''b''pb''b''ab''b''zb''b''yb'' ↩
       b''eb''b''tb''b''cb''b''yb'' b''cb''b''lb''b''eb''b''db''b''yb''b''ub''b''wb''b''ib'' ↩
       b''mb''
   b''ob''b''6b''b''pb''b''ab''b''zb''b''ob''b''mb'':
   [NAME]:[MAJOR].[MINOR].[PATCH].[REVISION]
   b''rb''b''db''b''eb'':
   NAME = b''ib''b''mb''b''yb'', b''pb''b''ob''b''lb''b''yb''b''cb''b''eb''b''nb''b''nb''b' ↩
       'ob''b''eb'' b''ib''b''zb'' GetKicadPluginClass()
   MAJOR, MINOR, PATCH, REVISION = b''ib''b''nb''b''fb''b''ob''b''pb''b''mb''b''ab''b''wb'' ↩
       b''ib''b''yb'' b''ob'' b''vb''b''eb''b''pb''b''cb''b''ib''b''ib'', b''pb''b''ob''b' ↩
       'lb''b''yb''b''cb''b''eb''b''nb''b''nb''b''ab''b''yb'' b''ib''b''zb''
   GetPluginVersion()
 */
```

```
void GetPluginInfo( std::string& aPluginInfo );
```

В общем случае, пользователь должен выполнить следующее:

1. Создать объект класса `KICAD_PLUGIN_LDR_3D`.
2. Вызвать функцию `Open( "/path/to/myplugin.so" )`, чтобы открыть нужный плагин. Возвращаемое значение нужно проверять, чтобы убедиться в успешной загрузке плагина.
3. Вызвать любую функцию из класса 3D-плагинов, обнаруженную в `KICAD_PLUGIN_LDR_3D`.
4. Вызвать `Close()` чтобы закрыть (выгрузить) плагин.
5. Удалить объект класса `KICAD_PLUGIN_LDR_3D`.

## 3.2 API калсса графа сцены

API класса графа сцены определён в заголовочном файле `ifsg_all.h` и вложенных в него. API содержит несколько дополнительных методов, объявленных в пространстве имён (namespace) `S3D` в файле `ifsg_api.h` и вспомогательных классов, объявленных в различных заголовочных файлах `ifsg_*.h`. Вспомогательные классы поддерживают основные форматы графов сцены, которые вместе образуют структуру графов, совместимую с VRML2.0. Заголовочные файлы, структуры, классы и их общедоступные функции рассмотрены далее:

**sg_version.h**

```
/*
   b''Ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''eb''b''нb''b''иb''b''eb'' b''иb''b' ↩
       'нb''b''фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''иb''b''иb'' b''ob'' b''вb''b''eb''b ↩
       ''pb''b''cb''b''иb''b''иb'' b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' b''гb''b''pb'' ↩
       b''ab''b''фb''b''ab'' b''cb''b''цb''b''eb''b''нb''b''ыb''.
   b''Bb''b''cb''b''eb'' b''пb''b''лb''b''ab''b''гb''b''иb''b''нb''b''ыb'', b''иb''b''cb''b ↩
       ''пb''b''ob''b''лb''b''ьb''b''зb''b''уb''b''юb''b''щb''b''иb''b''eb'' b''кb''b''лb''b ↩
       ''ab''b''cb''b''cb'' b''гb''b''pb''b''ab''b''фb''b''ab'' b''cb''b''цb''b''eb''b''нb'' ↩
       b''ыb'' b''дb''b''ob''b''лb''b''жb''b''нb''b''ыb'' b''вb''b''кb''b''лb''b''юb''b' ↩
       'чb''b''ab''b''тb''b''ьb'' b''эb''b''тb''b''ob''b''тb''
   b''зb''b''ab''b''гb''b''ob''b''лb''b''ob''b''вb''b''ob''b''чb''b''нb''b''ыb''b''йb'' b' ↩
       'фb''b''ab''b''йb''b''лb'' b''иb'' b''пb''b''pb''b''ob''b''вb''b''eb''b''pb''b''яb''b ↩
       ''тb''b''ьb'' b''вb''b''eb''b''pb''b''cb''b''иb''b''юb'' b''кb''b''ab''b''жb''b''дb'' ↩
       b''ыb''b''йb'' b''pb''b''ab''b''зb'', b''иb''b''cb''b''пb''b''ob''b''лb''b''ьb''b' ↩
       'зb''b''уb''b''яb'' b''pb''b''eb''b''зb''b''уb''b''лb''b''ьb''b''тb''b''ab''b''тb''
   b''фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''иb'' S3D::GetLibVersion(), b''дb''b''лb''b' ↩
       'яb'' b''пb''b''ob''b''дb''b''тb''b''вb''b''eb''b''pb''b''жb''b''дb''b''eb''b''нb''b' ↩
       'иb''b''яb'' b''cb''b''ob''b''вb''b''мb''b''eb''b''cb''b''тb''b''иb''b''мb''b''ob''b' ↩
       'cb''b''тb''b''иb''
 */

#define KICADSG_VERSION_MAJOR         2
#define KICADSG_VERSION_MINOR         0
#define KICADSG_VERSION_PATCH         0
#define KICADSG_VERSION_REVISION      0
```

**sg_types.h**

```
/*
   b''Ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''eb''b''нb''b''иb''b''eb'' b''тb''b' ↩
       'иb''b''пb''b''ob''b''вb'' b''дb''b''лb''b''яb'' b''кb''b''лb''b''ab''b''cb''b''cb''b ↩
       ''ab'' b''гb''b''pb''b''ab''b''фb''b''ab'' b''cb''b''цb''b''eb''b''нb''b''ыb''; b' ↩
       'эb''b''тb''b''иb'' b''тb''b''иb''b''пb''b''ыb''
   b''мb''b''ab''b''кb''b''cb''b''иb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''ob'' b''пb''b' ↩
       'pb''b''иb''b''бb''b''лb''b''иb''b''жb''b''eb''b''нb''b''ыb'' b''кb'' b''тb''b''иb''b ↩
       ''пb''b''ab''b''мb'' b''уb''b''зb''b''лb''b''ob''b''вb'' VRML2.0.
 */

namespace S3D
{
    enum SGTYPES
    {
        SGTYPE_TRANSFORM = 0,
        SGTYPE_APPEARANCE,
        SGTYPE_COLORS,
        SGTYPE_COLORINDEX,
        SGTYPE_FACESET,
        SGTYPE_COORDS,
        SGTYPE_COORDINDEX,
        SGTYPE_NORMALS,
        SGTYPE_SHAPE,
        SGTYPE_END
    };
};
```

Заголовочный файл `sg_base.h` состоит из объявлений основных типов данных, которые используются в классах графов сцены.

**sg_base.h**

```
/*
    b''Эb''b''тb''b''ab'' b''мb''b''ob''b''дb''b''eb''b''лb''b''ьb'' RGB-b''цb''b''вb''b' ↩
        'eb''b''тb''b''ab'' b''ab''b''нb''b''ab''b''лb''b''ob''b''гb''b''иb''b''чb''b''нb''b ↩
        ''ab'' b''мb''b''ob''b''дb''b''eb''b''лb''b''иb'' VRML2.0, b''гb''b''дb''b''eb'' b' ↩
        'кb''b''ab''b''жb''b''дb''b''ob''b''мb''b''уb''
    b''цb''b''вb''b''eb''b''тb''b''уb'' b''пb''b''pb''b''иb''b''cb''b''вb''b''ab''b''иb''b' ↩
        'вb''b''ab''b''eb''b''тb''b''cb''b''яb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b ↩
        ''иb''b''eb'' b''вb'' b''дb''b''иb''b''ab''b''пb''b''ab''b''зb''b''ob''b''нb''b' ↩
        'eb'' [0..1].
 */

class SGCOLOR
{
public:
    SGCOLOR();
```

```
    SGCOLOR( float aRVal, float aGVal, float aBVal );

    void GetColor( float& aRedVal, float& aGreenVal, float& aBlueVal ) const;
    void GetColor( SGCOLOR& aColor ) const;
    void GetColor( SGCOLOR* aColor ) const;

    bool SetColor( float aRedVal, float aGreenVal, float aBlueVal );
    bool SetColor( const SGCOLOR& aColor );
    bool SetColor( const SGCOLOR* aColor );
};


class SGPOINT
{
public:
    double x;
    double y;
    double z;

public:
    SGPOINT();
    SGPOINT( double aXVal, double aYVal, double aZVal );

    void GetPoint( double& aXVal, double& aYVal, double& aZVal );
    void GetPoint( SGPOINT& aPoint );
    void GetPoint( SGPOINT* aPoint );

    void SetPoint( double aXVal, double aYVal, double aZVal );
    void SetPoint( const SGPOINT& aPoint );
};


/*
    SGVECTOR b''ив''b''мв''b''еb''b''еb''b''тb'' 3 b''сb''b''оb''b''сb''b''тb''b''аb''b' ↩
        'вb''b''лb''b''яв''b''юв''b''щb''b''ив''b''еb'' (x,y,z), b''пb''b''оb''b''дb''b' ↩
        'оb''b''бв''b''нв''b''оb'' b''тb''b''оb''b''чb''b''кb''b''еb'', b''нв''b''оb''
    b''вb''b''еb''b''кb''b''тb''b''оb''b''рb'' b''сb''b''оb''b''дb''b''еb''b''рb''b''жb''b' ↩
        'ив''b''тb'' b''нв''b''оb''b''рb''b''мв''b''аb''b''лb''b''ив''b''зв''b''оb''b''вb''b ↩
        ''аb''b''нв''b''нв''b''ыb''b''еb'' b''зв''b''нв''b''аb''b''чb''b''еb''b''нв''b''ив'' ↩
        b''яв'' b''ив'' b''пв''b''рb''b''еb''b''дb''b''оb''b''тb''b''вb''b''рb''b''аb''b' ↩
        'щb''b''аb''b''еb''b''тb''
    b''ив''b''хb'' b''нв''b''еb''b''пb''b''оb''b''сb''b''рb''b''еb''b''дb''b''сb''b''тb''b' ↩
        'вb''b''еb''b''нв''b''нв''b''оb''b''еb'' b''ив''b''зв''b''мв''b''еb''b''нв''b''еb''b ↩
        ''нв''b''ив''b''еb''.
 */
class SGVECTOR
{
public:
```

```
    SGVECTOR();
    SGVECTOR( double aXVal, double aYVal, double aZVal );

    void GetVector( double& aXVal, double& aYVal, double& aZVal ) const;

    void SetVector( double aXVal, double aYVal, double aZVal );
    void SetVector( const SGVECTOR& aVector );

    SGVECTOR& operator=( const SGVECTOR& source );
};
```

Класс `IFSG_NODE` — базовый класс для всех узлов графа сцены. Все объекты графа сцены реализуют общедоступные функции этого класса, хотя не все они используются некоторыми объектами.

**ifsg_node.h**

```
class IFSG_NODE
{
public:
    IFSG_NODE();
    virtual ~IFSG_NODE();

    /**
     * b''Фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''яb'' Destroy
     * b''уb''b''дb''b''аb''b''лb''b''яb''b''еb''b''тb'' b''дb''b''аb''b''нb''b''нb''b' ↩
         'ыb''b''йb'' b''оb''b''бb''b''ъb''b''еb''b''кb''b''тb'' b''гb''b''рb''b''аb''b' ↩
         'фb''b''аb'' b''сb''b''цb''b''еb''b''нb''b''ыb''
     */
    void Destroy( void );

    /**
     * b''Фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''яb'' Attach
     * b''сb''b''вb''b''яb''b''зb''b''ыb''b''вb''b''аb''b''еb''b''тb'' b''пb''b''оb''b' ↩
         'лb''b''уb''b''чb''b''еb''b''нb''b''нb''b''ыb''b''йb'' SGNODE* b''сb'' b''эb''b' ↩
         'тb''b''иb''b''мb'' b''оb''b''бb''b''ъb''b''еb''b''кb''b''тb''b''оb''b''мb''
     */
    virtual bool Attach( SGNODE* aNode ) = 0;

    /**
     * b''Фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''яb'' NewNode
     * b''сb''b''оb''b''зb''b''дb''b''аb''b''ёb''b''тb'' b''нb''b''оb''b''вb''b''ыb''b' ↩
         'йb'' b''уb''b''зb''b''еb''b''лb'' b''иb'' b''сb''b''вb''b''яb''b''зb''b''ыb''b' ↩
         'вb''b''аb''b''еb''b''тb'' b''еb''b''гb''b''оb'' b''сb'' b''эb''b''тb''b''иb''b' ↩
         'мb'' b''оb''b''бb''b''ъb''b''еb''b''кb''b''тb''b''оb''b''мb''
     */
    virtual bool NewNode( SGNODE* aParent ) = 0;
    virtual bool NewNode( IFSG_NODE& aParent ) = 0;

    /**
```

```
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' GetRawPtr()
     * b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''yb''b' ↩
         'кb''b''ab''b''зb''b''ab''b''тb''b''eb''b''лb''b''ьb'' b''нb''b''eb''b''пb''b''ob'' ↩
         b''cb''b''pb''b''eb''b''дb''b''cb''b''тb''b''вb''b''eb''b''нb''b''нb''b''ob'' b' ↩
         'нb''b''ab'' SGNODE
     */
    SGNODE* GetRawPtr( void );

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' GetNodeType
     * b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''тb''b' ↩
         'иb''b''пb'' b''yb''b''зb''b''лb''b''ab'' b''дb''b''ab''b''нb''b''нb''b''ob''b' ↩
         'гb''b''ob'' b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb''b''ab''
     */
    S3D::SGTYPES GetNodeType( void ) const;

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' GetParent
     * b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''yb''b' ↩
         'кb''b''ab''b''зb''b''ab''b''тb''b''eb''b''лb''b''ьb'' b''нb''b''ab'' b''pb''b' ↩
         'ob''b''дb''b''иb''b''тb''b''eb''b''лb''b''ьb''b''cb''b''кb''b''иb''b''йb'' SGNODE ↩
         b''дb''b''лb''b''яb'' b''эb''b''тb''b''ob''b''гb''b''ob'' b''ob''b''бb''b''ъb''b' ↩
         'eb''b''кb''b''тb''b''ab''
     * b''иb''b''лb''b''иb'' NULL, b''eb''b''cb''b''лb''b''иb'' b''ob''b''бb''b''ъb''b' ↩
         'eb''b''кb''b''тb'' b''нb''b''eb'' b''иb''b''мb''b''eb''b''eb''b''тb'' b''pb''b' ↩
         'ob''b''дb''b''иb''b''тb''b''eb''b''лb''b''eb''b''йb'' (b''тb''.b''eb''. b''яb''b' ↩
         'вb''b''лb''b''яb''b''eb''b''тb''b''cb''b''яb''
     * b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb''b''ob''b''мb'' b''пb''b''pb''b''eb''b' ↩
         'ob''b''бb''b''pb''b''ab''b''зb''b''ob''b''вb''b''ab''b''нb''b''иb''b''яb'' b''вb'' ↩
         b''eb''b''pb''b''хb''b''нb''b''eb''b''гb''b''ob'' b''yb''b''pb''b''ob''b''вb''b' ↩
         'нb''b''яb'') b''иb''b''лb''b''иb'', b''кb''b''ob''b''гb''b''дb''b''ab'' b''дb''b' ↩
         'ab''b''нb''b''нb''b''ыb''b''йb''
     * b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb'' b''нb''b''eb'' b''cb''b''вb''b''яb''b' ↩
         'зb''b''ab''b''нb'' b''cb'' SGNODE.
     */
    SGNODE* GetParent( void ) const;

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' SetParent
     * b''пb''b''pb''b''иb''b''cb''b''вb''b''ab''b''иb''b''вb''b''ab''b''eb''b''тb'' b' ↩
         'pb''b''ob''b''дb''b''иb''b''тb''b''eb''b''лb''b''ьb''b''cb''b''кb''b''иb''b''йb'' ↩
         SGNODE b''дb''b''лb''b''яb'' b''дb''b''ab''b''нb''b''нb''b''ob''b''гb''b''ob'' b' ↩
         'ob''b''бb''b''ъb''b''eb''b''кb''b''тb''b''ab''.
     *
     * @param aParent [b''вb''b''хb''b''ob''b''дb''b''яb''b''щb''b''иb''b''йb''] b''жb''b' ↩
         'eb''b''лb''b''ab''b''eb''b''мb''b''ыb''b''йb'' b''pb''b''ob''b''дb''b''иb''b''тb'' ↩
         b''eb''b''лb''b''ьb'' b''yb''b''зb''b''лb''b''ab''
     * @return true b''eb''b''cb''b''лb''b''иb'' b''ob''b''пb''b''eb''b''pb''b''ab''b''цb'' ↩
```

```
        b''ив''b''яв'' b''вв''b''ыв''b''пв''b''ов''b''лв''b''нв''b''ев''b''нв''b''ав''; ↩
        false -
     * b''пв''b''ов''b''лв''b''ув''b''чв''b''ев''b''нв''b''нв''b''ыв''b''йв'' b''ув''b' ↩
        'зв''b''ев''b''лв'' b''нв''b''ев'' b''мв''b''ов''b''жв''b''ев''b''тв'' b''бв''b' ↩
        'ыв''b''тв''b''ьв'' b''рв''b''ов''b''дв''b''ив''b''тв''b''ев''b''лв''b''ев''b''мв'' ↩
         b''дв''b''лв''b''яв''
     * b''дв''b''ав''b''нв''b''нв''b''ов''b''гв''b''ов'' b''ов''b''бв''b''ъв''b''ев''b' ↩
        'кв''b''тв''b''ав''.
     */
    bool SetParent( SGNODE* aParent );

    /**
     * b''Фв''b''ув''b''нв''b''кв''b''цв''b''ив''b''яв'' GetNodeTypeName
     * b''вв''b''ов''b''зв''b''вв''b''рв''b''ав''b''щв''b''ав''b''ев''b''тв'' b''тв''b' ↩
        'ив''b''пв'' b''ув''b''зв''b''лв''b''ав'' b''вв'' b''вв''b''ив''b''дв''b''ев'' b' ↩
        'тв''b''ев''b''кв''b''св''b''тв''b''ав'' b''ив''b''лв''b''ив'' NULL, b''ев''b''св'' ↩
        b''лв''b''ив'' b''ув''b''зв''b''ев''b''лв'',
     * b''кв''b''ав''b''кв''b''ив''b''мв''-b''тв''b''ов'' b''ов''b''бв''b''рв''b''ав''b' ↩
        'зв''b''ов''b''мв'', b''ив''b''мв''b''ев''b''ев''b''тв'' b''нв''b''ев''b''вв''b' ↩
        'ев''b''рв''b''нв''b''ыв''b''йв'' b''тв''b''ив''b''пв''.
     */
    const char * GetNodeTypeName( S3D::SGTYPES aNodeType ) const;

    /**
     * b''Фв''b''ув''b''нв''b''кв''b''цв''b''ив''b''яв'' AddRefNode
     * b''дв''b''ов''b''бв''b''ав''b''вв''b''лв''b''яв''b''ев''b''тв'' b''св''b''св''b' ↩
        'ыв''b''лв''b''кв''b''ув'' b''нв''b''ав'' b''св''b''ув''b''щв''b''ев''b''св''b' ↩
        'тв''b''вв''b''ув''b''юв''b''щв''b''ив''b''йв'' b''ув''b''зв''b''ев''b''лв'', b' ↩
        'кв''b''ов''b''тв''b''ов''b''рв''b''ыв''b''йв'' b''нв''b''ев'' b''пв''b''рв''b' ↩
        'ив''b''нв''b''ав''b''дв''b''лв''b''ев''b''жв''b''ив''b''тв''
     * (b''нв''b''ев'' b''яв''b''вв''b''лв''b''яв''b''ев''b''тв''b''св''b''яв'' b''дв''b' ↩
        'ов''b''чв''b''ев''b''рв''b''нв''b''ив''b''мв'') b''эв''b''тв''b''ов''b''мв''b' ↩
        'ув'' b''ов''b''бв''b''ъв''b''ев''b''кв''b''тв''b''ув''.
     *
     * @return true b''пв''b''рв''b''ив'' b''ув''b''св''b''пв''b''ев''b''шв''b''нв''b''ов'' ↩
        b''мв'' b''зв''b''ав''b''вв''b''ев''b''рв''b''шв''b''ев''b''нв''b''ив''b''ив''
     */
    bool AddRefNode( SGNODE* aNode );
    bool AddRefNode( IFSG_NODE& aNode );

    /**
     * b''Фв''b''ув''b''нв''b''кв''b''цв''b''ив''b''яв'' AddChildNode
     * b''дв''b''ов''b''бв''b''ав''b''вв''b''лв''b''яв''b''ев''b''тв'' b''ув''b''зв''b' ↩
        'ев''b''лв'', b''яв''b''вв''b''лв''b''яв''b''юв''b''щв''b''ив''b''йв''b''св''b' ↩
        'яв'' b''дв''b''ов''b''чв''b''ев''b''рв''b''нв''b''ив''b''мв'' b''пв''b''ов'' b' ↩
        'ов''b''тв''b''нв''b''ов''b''шв''b''ев''b''нв''b''ив''b''юв'' b''кв'' b''эв''b' ↩
        'тв''b''ов''b''мв''b''ув'' b''ов''b''бв''b''ъв''b''ев''b''кв''b''тв''b''ув''.
     *
```

```
     * @return true b''пb''b''pb''b''иb'' b''yb''b''cb''b''пb''b''eb''b''шb''b''нb''b''ob'' ↩
         b''мb'' b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''eb''b''нb''b''иb''b''иb''
     */
    bool AddChildNode( SGNODE* aNode );
    bool AddChildNode( IFSG_NODE& aNode );
};
```

IFSG_TRANSFORM подобен узлу Transform из VRML2.0. Он может содержать любое количество дочерних или связанных узлов IFSG_SHAPE и IFSG_TRANSFORM. Корректный граф сцены должен иметь только один объект IFSG_TRANSFORM в качестве корневого.

**ifsg_transform.h**

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_TRANSFORM
 * b''эb''b''тb''b''ob'' b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ab'' b''дb''b' ↩
     'лb''b''яb'' b''cb''b''ob''b''вb''b''мb''b''eb''b''cb''b''тb''b''иb''b''мb''b''ob''b' ↩
     'cb''b''тb''b''иb'' b''cb'' b''бb''b''лb''b''ob''b''кb''b''ob''b''мb'' TRANSFORM b' ↩
     'иb''b''зb'' b''гb''b''pb''b''ab''b''фb''b''ab'' b''cb''b''цb''b''eb''b''нb''b''ыb''  ↩
     VRML
 */

class IFSG_TRANSFORM : public IFSG_NODE
{
public:
    IFSG_TRANSFORM( bool create );
    IFSG_TRANSFORM( SGNODE* aParent );

    bool SetScaleOrientation( const SGVECTOR& aScaleAxis, double aAngle );
    bool SetRotation( const SGVECTOR& aRotationAxis, double aAngle );
    bool SetScale( const SGPOINT& aScale );
    bool SetScale( double aScale );
    bool SetCenter( const SGPOINT& aCenter );
    bool SetTranslation( const SGPOINT& aTranslation );

    /* b''пb''b''pb''b''ob''b''чb''b''иb''b''eb'' b''фb''b''yb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''кb''b''лb'' ↩
       b''ab''b''cb''b''cb''b''ab'', b''кb''b''ob''b''тb''b''ob''b''pb''b''ыb''b''eb'' b' ↩
       'зb''b''дb''b''eb''b''cb''b''ьb'' b''нb''b''eb'' b''pb''b''ab''b''cb''b''cb''b''мb'' ↩
       b''ab''b''тb''b''pb''b''иb''b''вb''b''ab''b''юb''b''тb''b''cb''b''яb'' */
```

IFSG_SHAPE подобен узлу Shape из VRML2.0. Он должен содержать единственный дочерний узел FACESET или ссылку на него. Также, может содержать дочерний узел APPEARANCE или ссылку на него.

**ifsg_shape.h**

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_SHAPE
 * b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ab'' b''дb''b''лb''b''яb'' b''кb''b' ↩
     'лb''b''ab''b''cb''b''cb''b''ab'' SGSHAPE
```

```
 */

class IFSG_SHAPE : public IFSG_NODE
{
public:
    IFSG_SHAPE( bool create );
    IFSG_SHAPE( SGNODE* aParent );
    IFSG_SHAPE( IFSG_NODE& aParent );

    /* b''пb''b''pb''b''ob''b''чb''b''иb''b''eb'' b''фb''b''yb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''кb''b''лb'' ↩
        b''ab''b''cb''b''cb''b''ab'', b''кb''b''ob''b''тb''b''ob''b''pb''b''ыb''b''eb'' b' ↩
        'зb''b''дb''b''eb''b''cb''b''ьb'' b''нb''b''eb'' b''pb''b''ab''b''cb''b''cb''b''мb'' ↩
        b''ab''b''тb''b''pb''b''иb''b''вb''b''ab''b''юb''b''тb''b''cb''b''яb'' */
```

IFSG_APPEARANCE подобен узлу Appearance из VRML2.0, но на данный момент, он реализован в соответствии с узлом Appearance, содержащим узел Material.

**ifsg_appearance.h**

```
class IFSG_APPEARANCE : public IFSG_NODE
{
public:
    IFSG_APPEARANCE( bool create );
    IFSG_APPEARANCE( SGNODE* aParent );
    IFSG_APPEARANCE( IFSG_NODE& aParent );

    bool SetEmissive( float aRVal, float aGVal, float aBVal );
    bool SetEmissive( const SGCOLOR* aRGBColor );
    bool SetEmissive( const SGCOLOR& aRGBColor );

    bool SetDiffuse( float aRVal, float aGVal, float aBVal );
    bool SetDiffuse( const SGCOLOR* aRGBColor );
    bool SetDiffuse( const SGCOLOR& aRGBColor );

    bool SetSpecular( float aRVal, float aGVal, float aBVal );
    bool SetSpecular( const SGCOLOR* aRGBColor );
    bool SetSpecular( const SGCOLOR& aRGBColor );

    bool SetAmbient( float aRVal, float aGVal, float aBVal );
    bool SetAmbient( const SGCOLOR* aRGBColor );
    bool SetAmbient( const SGCOLOR& aRGBColor );

    bool SetShininess( float aShininess );
    bool SetTransparency( float aTransparency );

    /* b''пb''b''pb''b''ob''b''чb''b''иb''b''eb'' b''фb''b''yb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''кb''b''лb'' ↩
        b''ab''b''cb''b''cb''b''ab'' b''нb''b''eb'' b''пb''b''ob''b''кb''b''ab''b''зb''b' ↩
```

```
        'ab''b''нb''b''нb''b''ыb''b''eb'' b''зb''b''дb''b''eb''b''cb''b''ьb'' */

    /* b''cb''b''лb''b''eb''b''дb''b''уb''b''юb''b''щb''b''иb''b''eb'' b''фb''b''уb''b' ↩
        'нb''b''кb''b''цb''b''иb''b''иb'' b''нb''b''eb'' b''иb''b''cb''b''пb''b''оb''b''лb'' ↩
        b''ьb''b''зb''b''уb''b''юb''b''тb''b''cb''b''яb'' b''уb''b''зb''b''лb''b''ab''b' ↩
        'мb''b''иb'' Appearance
       b''иb'' b''мb''b''оb''b''гb''b''уb''b''тb'' b''вb''b''оb''b''зb''b''вb''b''рb''b' ↩
           'ab''b''щb''b''ab''b''тb''b''ьb'' b''кb''b''оb''b''дb'' b''оb''b''шb''b''иb''b' ↩
           'бb''b''кb''b''иb''

        bool AddRefNode( SGNODE* aNode );
        bool AddRefNode( IFSG_NODE& aNode );
        bool AddChildNode( SGNODE* aNode );
        bool AddChildNode( IFSG_NODE& aNode );
     */
};
```

IFSG_FACESET подобен узлу Geometry из VRML2.0, который содержит узел IndexedFaceSet. Он должен состоять из одного дочернего узла COORDS или ссылки на него, одного дочернего узла COORDINDEX и одного дочернего узла NORMALS или ссылки на него. Дополнительно, он может содержать дочерний узел COLORS или ссылку на него. Элементарные функции операций над векторами предназначены помочь пользователям в связывании этих векторов с поверхностями. Далее указаны некоторые отличия от VRML2.0:

1. Векторы всегда относятся к вершинам.
2. Цвета всегда присваиваются вершинам.
3. Набор индексов координат должен описывать только треугольные грани.

**ifsg_faceset.h**

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_FACESET
 * b''эb''b''тb''b''оb'' b''оb''b''бb''b''оb''b''лb''b''оb''b''чb''b''кb''b''ab'' b''дb''b' ↩
     'лb''b''яb'' b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' SGFACESET
 */

class IFSG_FACESET : public IFSG_NODE
{
public:
    IFSG_FACESET( bool create );
    IFSG_FACESET( SGNODE* aParent );
    IFSG_FACESET( IFSG_NODE& aParent );

    bool CalcNormals( SGNODE** aPtr );

    /* b''пb''b''рb''b''оb''b''чb''b''иb''b''eb'' b''фb''b''уb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''оb''b''вb''b''оb''b''гb''b''оb'' b''кb''b''лb'' ↩
       b''ab''b''cb''b''cb''b''ab'', b''кb''b''оb''b''тb''b''оb''b''рb''b''ыb''b''eb'' b' ↩
```

```
        'зb''b''дb''b''eb''b''cb''b''ьb'' b''нb''b''eb'' b''pb''b''ab''b''cb''b''cb''b''мb'' ↩
        b''ab''b''тb''b''pb''b''иb''b''вb''b''ab''b''юb''b''тb''b''cb''b''яb'' */
```

**ifsg_coords.h**

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_COORDS
 * b''эb''b''тb''b''ob'' b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ab'' b''дb''b' ↩
    'лb''b''яb'' SGCOORDS
 */

class IFSG_COORDS : public IFSG_NODE
{
public:
    IFSG_COORDS( bool create );
    IFSG_COORDS( SGNODE* aParent );
    IFSG_COORDS( IFSG_NODE& aParent );

    bool GetCoordsList( size_t& aListSize, SGPOINT*& aCoordsList );
    bool SetCoordsList( size_t aListSize, const SGPOINT* aCoordsList );
    bool AddCoord( double aXValue, double aYValue, double aZValue );
    bool AddCoord( const SGPOINT& aPoint );

    /* b''пb''b''pb''b''ob''b''чb''b''иb''b''eb'' b''фb''b''уb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''кb''b''лb'' ↩
       b''ab''b''cb''b''cb''b''ab'' b''нb''b''eb'' b''пb''b''ob''b''кb''b''ab''b''зb''b' ↩
        'ab''b''нb''b''нb''b''ыb''b''eb'' b''зb''b''дb''b''eb''b''cb''b''ьb'' */

    /* b''cb''b''лb''b''eb''b''дb''b''уb''b''юb''b''щb''b''иb''b''eb'' b''фb''b''уb''b' ↩
        'нb''b''кb''b''цb''b''иb''b''иb'' b''нb''b''eb'' b''иb''b''мb''b''eb''b''юb''b''тb'' ↩
         b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''яb'' b''дb''b''лb''b''яb'' b' ↩
        'уb''b''зb''b''лb''b''ob''b''вb''
       b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb'' b''иb'' b''вb''b' ↩
           'cb''b''eb''b''гb''b''дb''b''ab'' b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b' ↩
           'щb''b''ab''b''юb''b''тb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b' ↩
           'eb'' b''ob''b''шb''b''иb''b''бb''b''кb''b''иb''

        bool AddRefNode( SGNODE* aNode );
        bool AddRefNode( IFSG_NODE& aNode );
        bool AddChildNode( SGNODE* aNode );
        bool AddChildNode( IFSG_NODE& aNode );
     */
};
```

`IFSG_COORDINDEX` подобен массиву coordIdx[] из VRML2.0, он он должен описывать только стороны треугольников и, таким образом, общее количество индексов должно быть кратным 3-м.

**ifsg_coordindex.h**

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_COORDINDEX
 * b''эb''b''тb''b''ob'' b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ab'' b''дb''b' ↩
     'лb''b''яb'' SGCOORDINDEX
 */

class IFSG_COORDINDEX : public IFSG_INDEX
{
public:
    IFSG_COORDINDEX( bool create );
    IFSG_COORDINDEX( SGNODE* aParent );
    IFSG_COORDINDEX( IFSG_NODE& aParent );

    bool GetIndices( size_t& nIndices, int*& aIndexList );
    bool SetIndices( size_t nIndices, int* aIndexList );
    bool AddIndex( int aIndex );

    /* b''пb''b''pb''b''ob''b''чb''b''иb''b''eb'' b''фb''b''уb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''кb''b''лb'' ↩
       b''ab''b''cb''b''cb''b''ab'' b''нb''b''eb'' b''пb''b''ob''b''кb''b''ab''b''зb''b' ↩
        'ab''b''нb''b''нb''b''ыb''b''eb'' b''зb''b''дb''b''eb''b''cb''b''ьb'' */

    /* b''cb''b''лb''b''eb''b''дb''b''уb''b''юb''b''щb''b''иb''b''eb'' b''фb''b''уb''b' ↩
        'нb''b''кb''b''цb''b''иb''b''иb'' b''нb''b''eb'' b''иb''b''мb''b''eb''b''юb''b''тb'' ↩
         b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''яb'' b''дb''b''лb''b''яb'' b' ↩
        'уb''b''зb''b''лb''b''ab''
       b''иb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ob''b''вb'' b''кb''b''ob''b''ob''b' ↩
           'pb''b''дb''b''иb''b''нb''b''ab''b''тb'' b''иb'' b''вb''b''cb''b''eb''b''гb''b' ↩
           'дb''b''ab'' b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''юb''b' ↩
           'тb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''eb'' b''ob''b''шb''b' ↩
           'иb''b''бb''b''кb''b''иb''

        bool AddRefNode( SGNODE* aNode );
        bool AddRefNode( IFSG_NODE& aNode );
        bool AddChildNode( SGNODE* aNode );
        bool AddChildNode( IFSG_NODE& aNode );
     */
};
```

IFSG_NORMALS соответствует узлу Normals из VRML2.0.

**ifsg_normals.h**

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_NORMALS
 * b''эb''b''тb''b''ob'' b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ab'' b''дb''b' ↩
     'лb''b''яb'' b''кb''b''лb''b''ab''b''cb''b''cb''b''ab'' SGNORMALS
 */
```

```
class IFSG_NORMALS : public IFSG_NODE
{
public:
    IFSG_NORMALS( bool create );
    IFSG_NORMALS( SGNODE* aParent );
    IFSG_NORMALS( IFSG_NODE& aParent );

    bool GetNormalList( size_t& aListSize, SGVECTOR*& aNormalList );
    bool SetNormalList( size_t aListSize, const SGVECTOR* aNormalList );
    bool AddNormal( double aXValue, double aYValue, double aZValue );
    bool AddNormal( const SGVECTOR& aNormal );

    /* b''пb''b''pb''b''ob''b''чb''b''иb''b''eb'' b''фb''b''уb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''кb''b''лb'' ↩
       b''ab''b''cb''b''cb''b''ab'' b''нb''b''eb'' b''пb''b''ob''b''кb''b''ab''b''зb''b' ↩
        'ab''b''нb''b''нb''b''ыb''b''eb'' b''зb''b''дb''b''eb''b''cb''b''ьb'' */

    /* b''cb''b''лb''b''eb''b''дb''b''уb''b''юb''b''щb''b''иb''b''eb'' b''фb''b''уb''b' ↩
        'нb''b''кb''b''цb''b''иb''b''иb'' b''нb''b''eb'' b''иb''b''мb''b''eb''b''юb''b''тb'' ↩
         b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''яb'' b''дb''b''лb''b''яb'' b' ↩
        'уb''b''зb''b''лb''b''ab''
       b''вb''b''eb''b''кb''b''тb''b''ob''b''pb''b''ob''b''вb'' b''иb'' b''вb''b''cb''b' ↩
           'eb''b''гb''b''дb''b''ab'' b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b' ↩
           'ab''b''юb''b''тb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''eb'' b' ↩
           'ob''b''шb''b''иb''b''бb''b''кb''b''иb''

        bool AddRefNode( SGNODE* aNode );
        bool AddRefNode( IFSG_NODE& aNode );
        bool AddChildNode( SGNODE* aNode );
        bool AddChildNode( IFSG_NODE& aNode );
     */
};
```

IFSG_COLORS подобен массиву colors[] из VRML2.0.

**ifsg_colors.h**

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_COLORS
 * b''эb''b''тb''b''ob'' b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ab'' b''дb''b' ↩
     'лb''b''яb'' SGCOLORS
 */

class IFSG_COLORS : public IFSG_NODE
{
public:
    IFSG_COLORS( bool create );
    IFSG_COLORS( SGNODE* aParent );
```

```
    IFSG_COLORS( IFSG_NODE& aParent );

    bool GetColorList( size_t& aListSize, SGCOLOR*& aColorList );
    bool SetColorList( size_t aListSize, const SGCOLOR* aColorList );
    bool AddColor( double aRedValue, double aGreenValue, double aBlueValue );
    bool AddColor( const SGCOLOR& aColor );

    /* b''пb''b''pb''b''ob''b''чb''b''иb''b''eb'' b''фb''b''уb''b''нb''b''кb''b''цb''b' ↩
        'иb''b''иb'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''кb''b''лb'' ↩
       b''ab''b''cb''b''cb''b''ab'' b''нb''b''eb'' b''пb''b''ob''b''кb''b''ab''b''зb''b' ↩
        'ab''b''нb''b''нb''b''ыb''b''eb'' b''зb''b''дb''b''eb''b''cb''b''ьb'' */

    /* b''cb''b''лb''b''eb''b''дb''b''уb''b''юb''b''щb''b''иb''b''eb'' b''фb''b''уb''b' ↩
        'нb''b''кb''b''цb''b''иb''b''иb'' b''нb''b''eb'' b''иb''b''мb''b''eb''b''юb''b''тb'' ↩
         b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''яb'' b''дb''b''лb''b''яb'' b' ↩
        'уb''b''зb''b''лb''b''ab''
       b''вb''b''eb''b''кb''b''тb''b''ob''b''pb''b''ob''b''вb'' b''иb'' b''вb''b''cb''b' ↩
           'eb''b''гb''b''дb''b''ab'' b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b' ↩
           'ab''b''юb''b''тb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''eb'' b' ↩
           'ob''b''шb''b''иb''b''бb''b''кb''b''иb''

        bool AddRefNode( SGNODE* aNode );
        bool AddRefNode( IFSG_NODE& aNode );
        bool AddChildNode( SGNODE* aNode );
        bool AddChildNode( IFSG_NODE& aNode );
     */
};
```

Остальные функции API определены в `ifsg_api.h` и показаны далее:

**ifsg_api.h**

```
namespace S3D
{
    /**
     * b''Фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''яb'' GetLibVersion b''вb''b''ob''b' ↩
         'зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''иb''b''нb''b''фb''b''ob'' ↩
        b''pb''b''мb''b''ab''b''цb''b''иb''b''юb'' b''ob'' b''вb''b''eb''b''pb''b''cb''b' ↩
         'иb''b''иb''
     * b''бb''b''иb''b''бb''b''лb''b''иb''b''ob''b''тb''b''eb''b''кb''b''иb'' kicad_3dsg
     */
    SGLIB_API void GetLibVersion( unsigned char* Major, unsigned char* Minor,
                                  unsigned char* Patch, unsigned char* Revision );

    // b''фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''иb'' b''дb''b''лb''b''яb'' b''иb''b' ↩
        'зb''b''вb''b''лb''b''eb''b''чb''b''eb''b''нb''b''иb''b''яb'' b''иb''b''нb''b''фb''b ↩
        ''ob''b''pb''b''мb''b''ab''b''цb''b''иb''b''иb'' b''пb''b''ob'' b''уb''b''кb''b' ↩
        'ab''b''зb''b''ab''b''тb''b''eb''b''лb''b''яb''b''мb'' SGNODE
    SGLIB_API S3D::SGTYPES GetSGNodeType( SGNODE* aNode );
```

```
SGLIB_API SGNODE* GetSGNodeParent( SGNODE* aNode );
SGLIB_API bool AddSGNodeRef( SGNODE* aParent, SGNODE* aChild );
SGLIB_API bool AddSGNodeChild( SGNODE* aParent, SGNODE* aChild );
SGLIB_API void AssociateSGNodeWrapper( SGNODE* aObject, SGNODE** aRefPtr );

/**
 * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' CalcTriNorm
 * b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''нb''b' ↩
    'ob''b''pb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''ыb''b''йb'' b''вb''b''eb''b''кb'' ↩
    b''тb''b''ob''b''pb'' b''дb''b''лb''b''яb'' b''тb''b''pb''b''eb''b''yb''b''гb''b' ↩
    'ob''b''лb''b''ьb''b''нb''b''иb''b''кb''b''ab'', b''ob''b''пb''b''иb''b''cb''b' ↩
    'ab''b''нb''b''нb''b''ob''b''гb''b''ob'' b''вb''b''eb''b''pb''b''шb''b''иb''b''нb'' ↩
    b''ab''b''мb''b''иb'' p1, p2, p3
 */
SGLIB_API SGVECTOR CalcTriNorm( const SGPOINT& p1, const SGPOINT& p2, const SGPOINT& p3 ↩
    );

/**
 * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' WriteCache
 * b''зb''b''ab''b''пb''b''иb''b''cb''b''ыb''b''вb''b''ab''b''eb''b''тb'' b''дb''b' ↩
    'eb''b''pb''b''eb''b''вb''b''ob'' SGNODE b''вb'' b''бb''b''иb''b''нb''b''ab''b' ↩
    'pb''b''нb''b''ыb''b''йb'' b''фb''b''ab''b''йb''b''лb'' b''кb''b''эb''b''шb''b' ↩
    'ab''
 *
 * @param aFileName - b''нb''b''ab''b''зb''b''вb''b''ab''b''нb''b''иb''b''eb'' b''фb''b ↩
    ''ab''b''йb''b''лb''b''ab'' b''дb''b''лb''b''яb'' b''зb''b''ab''b''пb''b''иb''b' ↩
    'cb''b''иb''
 * @param overwrite - b''дb''b''ob''b''лb''b''жb''b''eb''b''нb'' b''cb''b''ob''b''дb''b ↩
    ''eb''b''pb''b''жb''b''ab''b''тb''b''ьb'' b''иb''b''cb''b''тb''b''иb''b''нb''b' ↩
    'yb'', b''eb''b''cb''b''лb''b''иb'' b''нb''b''yb''b''жb''b''нb''b''ob'' b''пb''b' ↩
    'eb''b''pb''b''eb''b''зb''b''ab''b''пb''b''иb''b''cb''b''ab''b''тb''b''ьb'' b''cb'' ↩
    b''yb''b''щb''b''eb''b''cb''b''тb''b''вb''b''yb''b''юb''b''щb''b''иb''b''йb'' b' ↩
    'фb''b''ab''b''йb''b''лb''
 * @param aNode - b''лb''b''юb''b''бb''b''ob''b''йb'' b''yb''b''зb''b''eb''b''лb'' b' ↩
    'иb''b''зb'' b''дb''b''eb''b''pb''b''eb''b''вb''b''ab'', b''кb''b''ob''b''тb''b' ↩
    'ob''b''pb''b''ьb''b''йb'' b''нb''b''yb''b''жb''b''нb''b''ob'' b''зb''b''ab''b' ↩
    'пb''b''иb''b''cb''b''ab''b''тb''b''ьb''
 * @return true b''пb''b''pb''b''иb'' b''yb''b''cb''b''пb''b''eb''b''шb''b''нb''b''ob'' ↩
    b''мb'' b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''eb''b''нb''b''иb''b''иb''
 */
SGLIB_API bool WriteCache( const char* aFileName, bool overwrite, SGNODE* aNode,
    const char* aPluginInfo );

/**
 * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' ReadCache
 * b''cb''b''чb''b''иb''b''тb''b''ыb''b''вb''b''ab''b''eb''b''тb'' b''бb''b''иb''b' ↩
    'нb''b''ab''b''pb''b''нb''b''ыb''b''йb'' b''фb''b''ab''b''йb''b''лb'' b''кb''b' ↩
    'эb''b''шb''b''ab'' b''иb'' b''cb''b''ob''b''зb''b''дb''b''ab''b''eb''b''тb'' b' ↩
```

```
        'дb''b''eb''b''pb''b''eb''b''вb''b''ob'' SGNODE
     *
     * @param aFileName - b''иb''b''мb''b''яb'' b''бb''b''иb''b''нb''b''ab''b''pb''b''нb''b ↩
        ''ob''b''гb''b''ob'' b''фb''b''ab''b''йb''b''лb''b''ab'' b''кb''b''эb''b''шb''b' ↩
        'ab'' b''дb''b''лb''b''яb'' b''cb''b''чb''b''иb''b''тb''b''ыb''b''вb''b''ab''b' ↩
        'нb''b''иb''b''яb''
     * @return NULL b''пb''b''pb''b''иb'' b''cb''b''бb''b''ob''b''eb'', b''вb'' b''cb''b' ↩
        'лb''b''уb''b''чb''b''ab''b''eb'' b''уb''b''cb''b''пb''b''eb''b''хb''b''ab'' - b' ↩
        'вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b''уb''b''кb'' ↩
        b''ab''b''зb''b''ab''b''тb''b''eb''b''лb''b''ьb'' b''нb''b''ab''
     * b''уb''b''зb''b''eb''b''лb'' b''вb''b''eb''b''pb''b''хb''b''нb''b''eb''b''гb''b' ↩
        'ob'' b''уb''b''pb''b''ob''b''вb''b''нb''b''яb'' SCENEGRAPH;
     * b''eb''b''cb''b''лb''b''иb'' b''пb''b''ob''b''нb''b''ab''b''дb''b''ob''b''бb''b' ↩
        'иb''b''тb''b''cb''b''яb'', b''эb''b''тb''b''ob''b''тb'' b''уb''b''зb''b''eb''b' ↩
        'лb'' b''мb''b''ob''b''жb''b''нb''b''ob'' b''cb''b''вb''b''яb''b''зb''b''ab''b' ↩
        'тb''b''ьb'' b''cb'' b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ob''b' ↩
        'йb'' IFSG_TRANSFORM
     * b''cb'' b''пb''b''ob''b''мb''b''ob''b''щb''b''ьb''b''юb'' b''фb''b''уb''b''нb''b' ↩
        'кb''b''цb''b''иb''b''иb'' IFSG_TRANSFORM::Attach().
     */
    SGLIB_API SGNODE* ReadCache( const char* aFileName, void* aPluginMgr,
        bool (*aTagCheck)( const char*, void* ) );

    /**
     * b''Фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''яb'' WriteVRML
     * b''зb''b''ab''b''пb''b''иb''b''cb''b''ыb''b''вb''b''ab''b''eb''b''тb'' b''пb''b' ↩
        'eb''b''pb''b''eb''b''дb''b''ab''b''нb''b''нb''b''ыb''b''йb'' b''уb''b''зb''b''eb'' ↩
        b''лb'' b''иb'' b''eb''b''гb''b''ob'' b''дb''b''ob''b''чb''b''eb''b''pb''b''нb''b' ↩
        'иb''b''eb'' b''уb''b''зb''b''лb''b''ыb'' b''вb'' b''фb''b''ab''b''йb''b''лb'' ↩
        VRML2
     *
     * @param filename - b''иb''b''мb''b''яb'' b''фb''b''ab''b''йb''b''лb''b''ab'' b''дb''b ↩
        ''лb''b''яb'' b''зb''b''ab''b''пb''b''иb''b''cb''b''иb''
     * @param overwrite - b''дb''b''ob''b''лb''b''жb''b''eb''b''нb'' b''бb''b''ыb''b''тb''b ↩
        ''ьb'' b''уb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb'' b' ↩
        'вb'' b''иb''b''cb''b''тb''b''иb''b''нb''b''уb'', b''чb''b''тb''b''ob''b''бb''b' ↩
        'ыb'' b''пb''b''eb''b''pb''b''eb''b''зb''b''ab''b''пb''b''иb''b''cb''b''ab''b''тb'' ↩
        b''ьb''
     * b''cb''b''уb''b''щb''b''eb''b''cb''b''тb''b''вb''b''уb''b''юb''b''щb''b''иb''b''йb'' ↩
         b''фb''b''ab''b''йb''b''лb'' VRML
     * @param aTopNode - b''уb''b''кb''b''ab''b''зb''b''ab''b''тb''b''eb''b''лb''b''ьb'' b' ↩
        'нb''b''ab'' b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb'' SCENEGRAPH, b''пb''b''pb''b ↩
        ''eb''b''дb''b''cb''b''тb''b''ab''b''вb''b''лb''b''яb''b''юb''b''щb''b''иb''b''йb'' ↩
         b''cb''b''цb''b''eb''b''нb''b''уb'' VRML
     * @param reuse - b''дb''b''ob''b''лb''b''жb''b''eb''b''нb'' b''бb''b''ыb''b''тb''b' ↩
        'ьb'' b''уb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb'' b' ↩
        'вb'' b''иb''b''cb''b''тb''b''иb''b''нb''b''уb'', b''дb''b''лb''b''яb'' b''иb''b' ↩
        'cb''b''пb''b''ob''b''лb''b''ьb''b''зb''b''ob''b''вb''b''ab''b''нb''b''иb''b''яb''
```

```
     * b''cb''b''вb''b''ob''b''йb''b''cb''b''тb''b''вb'' VRML DEF/USE
     * @return true b''пb''b''pb''b''иb'' b''yb''b''cb''b''пb''b''eb''b''шb''b''нb''b''ob'' ↩
        b''мb'' b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''eb''b''нb''b''иb''b''иb''
     */
    SGLIB_API bool WriteVRML( const char* filename, bool overwrite, SGNODE* aTopNode,
                    bool reuse, bool renameNodes );

    // b''Пb''b''Pb''b''Иb''b''Мb''b''Eb''b''Чb''b''Ab''b''Нb''b''Иb''b''Eb'': b''cb''b' ↩
        'лb''b''eb''b''дb''b''yb''b''юb''b''щb''b''иb''b''eb'' b''фb''b''yb''b''нb''b''кb''b ↩
        ''цb''b''иb''b''иb'' b''иb''b''cb''b''пb''b''ob''b''лb''b''ьb''b''зb''b''yb''b''юb'' ↩
        b''тb''b''cb''b''яb'' b''cb''b''ob''b''вb''b''мb''b''eb''b''cb''b''тb''b''нb''b' ↩
        'ob'' b''дb''b''лb''b''яb'' b''cb''b''ob''b''зb''b''дb''b''ab''b''нb''b''иb''b''яb'' ↩
         b''cb''b''бb''b''ob''b''pb''b''кb''b''иb'' VRML,
    // b''кb''b''ob''b''тb''b''ob''b''pb''b''ab''b''яb'' b''мb''b''ob''b''жb''b''eb''b' ↩
        'тb'' b''иb''b''cb''b''пb''b''ob''b''лb''b''ьb''b''зb''b''ob''b''вb''b''ab''b''тb''b ↩
        ''ьb'' b''нb''b''eb''b''cb''b''кb''b''ob''b''лb''b''ьb''b''кb''b''ob'' b''ob''b' ↩
        'бb''b''ъb''b''eb''b''кb''b''тb''b''ob''b''вb'' b''дb''b''лb''b''яb'' b''кb''b''ab'' ↩
        b''жb''b''дb''b''ob''b''гb''b''ob'' SG*-b''кb''b''лb''b''ab''b''cb''b''cb''b''ab''.
    // b''Bb'' b''ob''b''бb''b''ыb''b''чb''b''нb''b''ob''b''мb'' b''cb''b''лb''b''yb''b' ↩
        'чb''b''ab''b''eb'' b''дb''b''ob''b''лb''b''жb''b''нb''b''ob'' b''бb''b''ыb''b''тb'' ↩
        b''ьb'' b''тb''b''ab''b''кb'':
    // 1) b''вb''b''ыb''b''зb''b''ob''b''вb'' b''фb''b''yb''b''нb''b''кb''b''цb''b''иb''b' ↩
        'иb'' 'ResetNodeIndex()' b''дb''b''лb''b''яb'' b''cb''b''бb''b''pb''b''ob''b''cb''b' ↩
        'ab'' b''гb''b''лb''b''ob''b''бb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' ↩
        b''иb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ab'' b''иb''b''мb''b''eb''b''нb'' b' ↩
        'yb''b''зb''b''лb''b''ob''b''вb'';
    // 2) b''дb''b''лb''b''яb'' b''кb''b''ab''b''жb''b''дb''b''ob''b''гb''b''ob'' b''yb''b' ↩
        'кb''b''ab''b''зb''b''ab''b''тb''b''eb''b''лb''b''яb'' b''мb''b''ob''b''дb''b''eb''b ↩
        ''лb''b''иb'', b''пb''b''ob''b''лb''b''yb''b''чb''b''eb''b''нb''b''нb''b''ob''b' ↩
        'гb''b''ob'' b''cb'' b''пb''b''ob''b''мb''b''ob''b''щb''b''ьb''b''юb'' 'S3D_CACHE-> ↩
        Load()',
    //    b''eb''b''дb''b''иb''b''нb''b''ob''b''жb''b''дb''b''ыb'' b''вb''b''ыb''b''зb''b' ↩
        'ыb''b''вb''b''ab''b''eb''b''тb''b''cb''b''яb'' 'RenameNodes()'. b''Tb''b''ab''b' ↩
        'кb''b''иb''b''мb'' b''ob''b''бb''b''pb''b''ab''b''зb''b''ob''b''мb'' b''дb''b''ob'' ↩
        b''cb''b''тb''b''иb''b''гb''b''ab''b''юb''b''тb'' b''тb''b''ob''b''гb''b''ob'', b' ↩
        'чb''b''тb''b''ob''b''бb''b''ыb''
    //    b''вb''b''cb''b''eb'' b''yb''b''зb''b''лb''b''ыb'', b''пb''b''ob''b''лb''b''yb''b ↩
        ''чb''b''eb''b''нb''b''нb''b''ыb''b''eb'' b''иb''b''зb'' b''вb''b''ыb''b''хb''b' ↩
        'ob''b''дb''b''нb''b''ob''b''гb''b''ob'' b''фb''b''ab''b''йb''b''лb''b''ab'', b' ↩
        'иb''b''мb''b''eb''b''лb''b''иb'' b''yb''b''нb''b''иb''b''кb''b''ab''b''лb''b''ьb''b ↩
        ''нb''b''ыb''b''eb'' b''иb''b''мb''b''eb''b''нb''b''ab''.
    //    b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' RenameNodes() b''пb''b''eb''b' ↩
        'pb''b''eb''b''иb''b''мb''b''eb''b''нb''b''ob''b''вb''b''ыb''b''вb''b''ab''b''eb''b' ↩
        'тb'' b''пb''b''ob''b''лb''b''yb''b''чb''b''eb''b''нb''b''нb''b''ыb''b''йb'' b''yb'' ↩
        b''зb''b''eb''b''лb'' b''иb'' b''вb''b''cb''b''eb'' b''eb''b''гb''b''ob'' b''дb''b' ↩
        'ob''b''чb''b''eb''b''pb''b''нb''b''иb''b''eb''
    //    b''yb''b''зb''b''лb''b''ыb''. b''Cb''b''вb''b''яb''b''зb''b''ab''b''нb''b''нb''b' ↩
        'ыb''b''eb'' b''yb''b''зb''b''лb''b''ыb'' b''ob''b''cb''b''тb''b''ab''b''юb''b''тb'' ↩
```

```
    b''cb''b''яb'' b''бb''b''eb''b''зb'' b''иb''b''зb''b''мb''b''eb''b''нb''b''eb''b' ↩
    'нb''b''иb''b''йb''. b''Иb''b''cb''b''пb''b''оb''b''лb''b''ьb''b''зb''b''оb''b''вb'' ↩
    b''аb''b''нb''b''иb''b''eb'' b''уb''b''кb''b''аb''b''зb''b''аb''b''тb''b''eb''b' ↩
    'лb''b''яb'',
//    b''пb''b''оb''b''лb''b''уb''b''чb''b''eb''b''нb''b''нb''b''оb''b''гb''b''оb'' b' ↩
    'иb''b''зb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''иb'' 'S3DCACHE->Load()', b ↩
    ''пb''b''оb''b''зb''b''вb''b''оb''b''лb''b''яb''b''eb''b''тb'' b''уb''b''бb''b''eb'' ↩
    b''дb''b''иb''b''тb''b''ьb''b''cb''b''яb'' b''вb'' b''тb''b''оb''b''мb'', b''чb''b' ↩
    'тb''b''оb''
//    b''вb''b''cb''b''eb'' b''дb''b''оb''b''чb''b''eb''b''рb''b''нb''b''иb''b''eb'' b' ↩
    'уb''b''зb''b''лb''b''ыb'', b''пb''b''оb'' b''оb''b''тb''b''нb''b''оb''b''шb''b' ↩
    'eb''b''нb''b''иb''b''юb'' b''кb'' b''пb''b''оb''b''cb''b''лb''b''eb''b''дb''b''нb'' ↩
    b''eb''b''мb''b''уb'', b''бb''b''уb''b''дb''b''уb''b''тb'' b''иb''b''мb''b''eb''b' ↩
    'тb''b''ьb'' b''уb''b''нb''b''иb''b''кb''b''аb''b''лb''b''ьb''b''нb''b''ыb''b''eb'' ↩
    b''иb''b''мb''b''eb''b''нb''b''аb'';
// 3) b''eb''b''cb''b''лb''b''иb'' SG*-b''дb''b''eb''b''рb''b''eb''b''вb''b''оb'' b' ↩
    'cb''b''оb''b''зb''b''дb''b''аb''b''нb''b''оb'' b''нb''b''eb''b''зb''b''аb''b''вb''b ↩
    ''иb''b''cb''b''иb''b''мb''b''оb'' b''оb''b''тb'' S3DCACHE->Load(), b''тb''b''оb'' b ↩
    ''пb''b''оb''b''лb''b''ьb''b''зb''b''оb''b''вb''b''аb''b''тb''b''eb''b''лb''b''ьb''
//    b''дb''b''оb''b''лb''b''жb''b''eb''b''нb'' b''вb''b''ыb''b''зb''b''вb''b''аb''b' ↩
    'тb''b''ьb'' RenameNodes() b''кb''b''аb''b''кb'' b''пb''b''оb''b''лb''b''оb''b''жb'' ↩
    b''eb''b''нb''b''оb'', b''чb''b''тb''b''оb''b''бb''b''ыb'' b''оb''b''бb''b''eb''b' ↩
    'cb''b''пb''b''eb''b''чb''b''иb''b''тb''b''ьb'' b''вb''b''cb''b''eb'' b''уb''b''зb'' ↩
    b''лb''b''ыb''
//    b''уb''b''нb''b''иb''b''кb''b''аb''b''лb''b''ьb''b''нb''b''ыb''b''мb''b''иb'' b' ↩
    'иb''b''мb''b''eb''b''нb''b''аb''b''мb''b''иb'';
// 4) b''cb''b''оb''b''зb''b''дb''b''аb''b''тb''b''ьb'' b''cb''b''тb''b''рb''b''уb''b' ↩
    'кb''b''тb''b''уb''b''рb''b''уb'' b''cb''b''бb''b''оb''b''рb''b''кb''b''иb'' b''пb'' ↩
    b''уb''b''тb''b''ёb''b''мb'' b''cb''b''оb''b''зb''b''дb''b''аb''b''нb''b''иb''b' ↩
    'яb'' b''нb''b''оb''b''вb''b''оb''b''гb''b''оb'' b''уb''b''зb''b''лb''b''аb'' ↩
    IFSG_TRANSFORM, b''кb''b''аb''b''кb''
//    b''пb''b''оb''b''лb''b''аb''b''гb''b''аb''b''eb''b''тb''b''cb''b''яb'' b''дb''b' ↩
    'лb''b''яb'' b''кb''b''аb''b''жb''b''дb''b''оb''b''гb''b''оb'' b''эb''b''кb''b''зb'' ↩
    b''eb''b''мb''b''пb''b''лb''b''яb''b''рb''b''аb'' b''кb''b''оb''b''мb''b''пb''b' ↩
    'оb''b''нb''b''eb''b''нb''b''тb''b''оb''b''вb''; b''бb''b''аb''b''зb''b''оb''b''вb'' ↩
    b''уb''b''юb'' b''мb''b''оb''b''дb''b''eb''b''лb''b''ьb'' b''кb''b''оb''b''мb''b' ↩
    'пb''b''оb''b''нb''b''eb''b''нb''b''тb''b''аb'',
//    b''вb''b''оb''b''зb''b''вb''b''рb''b''аb''b''щb''b''аb''b''eb''b''мb''b''уb''b' ↩
    'юb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''иb''b''eb''b''йb'' S3DCACHE->Load(), b' ↩
    'мb''b''оb''b''жb''b''нb''b''оb'' b''дb''b''оb''b''бb''b''аb''b''вb''b''иb''b''тb''b ↩
    ''ьb'' b''кb'' b''дb''b''аb''b''нb''b''нb''b''оb''b''мb''b''уb'' b''уb''b''зb''b' ↩
    'лb''b''уb''
//    IFSG_TRANSFORM b''cb'' b''пb''b''оb''b''мb''b''оb''b''щb''b''ьb''b''юb'' ' ↩
    AddRefNode()';
// 5) b''уb''b''бb''b''eb''b''дb''b''иb''b''тb''b''ьb''b''cb''b''яb'', b''чb''b''тb''b' ↩
    'оb'' b''вb''b''cb''b''eb'' b''нb''b''оb''b''вb''b''ыb''b''eb'' b''уb''b''зb''b' ↩
    'лb''b''ыb'' IFSG_TRANSFORM b''дb''b''оb''b''бb''b''аb''b''вb''b''лb''b''eb''b''нb'' ↩
    b''ыb'' b''вb'' b''кb''b''аb''b''чb''b''eb''b''cb''b''тb''b''вb''b''eb'' b''дb''b' ↩
```

```
    'ob''b''чb''b''eb''b''pb''b''нb''b''иb''b''xb''
//    b''кb'' b''yb''b''зb''b''лb''b''yb'' b''вb''b''eb''b''pb''b''xb''b''нb''b''eb''b' ↩
    'гb''b''ob'' b''yb''b''pb''b''ob''b''вb''b''нb''b''яb'' IFSG_TRANSFORM, b''пb''b' ↩
    'ob''b''дb''b''гb''b''ob''b''тb''b''ob''b''вb''b''иb''b''вb'' b''eb''b''гb''b''ob'', ↩
     b''тb''b''ab''b''кb''b''иb''b''мb'' b''ob''b''бb''b''pb''b''ab''b''зb''b''ob''b' ↩
    'мb'', b''кb''
//    b''дb''b''ab''b''лb''b''ьb''b''нb''b''eb''b''йb''b''шb''b''eb''b''мb''b''yb'' b' ↩
    'пb''b''eb''b''pb''b''eb''b''иb''b''мb''b''eb''b''нb''b''ob''b''вb''b''ab''b''нb''b' ↩
    'иb''b''юb'' b''иb'' b''зb''b''ab''b''пb''b''иb''b''cb''b''иb'';
// 6) b''вb''b''ыb''b''зb''b''вb''b''ab''b''тb''b''ьb'' RenameNodes() b''дb''b''лb''b' ↩
    'яb'' b''yb''b''зb''b''лb''b''ab'' b''cb''b''бb''b''ob''b''pb''b''кb''b''иb'' b' ↩
    'вb''b''eb''b''pb''b''xb''b''нb''b''eb''b''гb''b''ob'' b''yb''b''pb''b''ob''b''вb''b ↩
    ''нb''b''яb'';
// 7) b''вb''b''ыb''b''зb''b''вb''b''ab''b''тb''b''ьb'' WriteVRML() b''вb'' b''ob''b' ↩
    'бb''b''ыb''b''чb''b''нb''b''ob''b''мb'' b''пb''b''ob''b''pb''b''яb''b''дb''b''кb''b ↩
    ''eb'', b''cb'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''ob''b' ↩
    'мb'' renameNodes = false,
//    b''чb''b''тb''b''ob''b''бb''b''ыb'' b''зb''b''ab''b''пb''b''иb''b''cb''b''ab''b' ↩
    'тb''b''ьb'' b''вb''b''cb''b''юb'' b''cb''b''тb''b''pb''b''yb''b''кb''b''тb''b''yb'' ↩
    b''pb''b''yb'' b''cb''b''бb''b''ob''b''pb''b''кb''b''иb'' b''вb'' b''ob''b''дb''b' ↩
    'иb''b''нb'' VRML-b''фb''b''ab''b''йb''b''лb''.
// 8) b''вb''b''ыb''b''cb''b''вb''b''ob''b''бb''b''ob''b''дb''b''иb''b''тb''b''ьb'' b' ↩
    'пb''b''ab''b''мb''b''яb''b''тb''b''ьb'', b''yb''b''дb''b''ab''b''лb''b''иb''b''вb'' ↩
     b''вb''b''cb''b''eb'' IFSG_TRANSFORM b''пb''b''eb''b''pb''b''eb''b''мb''b''eb''b' ↩
    'нb''b''нb''b''ыb''b''eb'' b''иb'' b''ob''b''бb''b''ъb''b''eb''b''кb''b''тb''b''ыb'' ↩
     b''пb''b''pb''b''ob''b''чb''b''иb''b''xb''
//    SG*-b''кb''b''лb''b''ab''b''cb''b''cb''b''ob''b''вb'', b''кb''b''ob''b''тb''b' ↩
    'ob''b''pb''b''ыb''b''eb'' b''бb''b''ыb''b''лb''b''иb'' b''cb''b''ob''b''зb''b''дb'' ↩
    b''ab''b''нb''b''ыb'' b''иb''b''cb''b''кb''b''лb''b''юb''b''чb''b''иb''b''тb''b' ↩
    'eb''b''лb''b''ьb''b''нb''b''ob'' b''дb''b''лb''b''яb'' b''зb''b''ab''b''пb''b''иb'' ↩
    b''cb''b''иb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''xb''.

/**
 * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' ResetNodeIndex
 * b''cb''b''бb''b''pb''b''ab''b''cb''b''ыb''b''вb''b''ab''b''eb''b''тb'' b''гb''b' ↩
     'лb''b''ob''b''бb''b''ab''b''лb''b''ьb''b''нb''b''ыb''b''eb'' b''иb''b''нb''b''дb'' ↩
     b''eb''b''кb''b''cb''b''ыb'' SG*-b''кb''b''лb''b''ab''b''cb''b''cb''b''ab''
 *
 * @param aNode - b''мb''b''ob''b''жb''b''eb''b''тb'' b''бb''b''ыb''b''тb''b''ьb'' b' ↩
     'лb''b''юb''b''бb''b''ыb''b''мb'' b''пb''b''ob''b''дb''b''xb''b''ob''b''дb''b''яb'' ↩
     b''щb''b''иb''b''мb'' SGNODE
 */
SGLIB_API void ResetNodeIndex( SGNODE* aNode );

/**
 * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' RenameNodes
 * b''пb''b''eb''b''pb''b''eb''b''иb''b''мb''b''eb''b''нb''b''ob''b''вb''b''ыb''b''вb'' ↩
     b''ab''b''eb''b''тb'' b''yb''b''зb''b''eb''b''лb'' b''иb'' b''eb''b''гb''b''ob'' b' ↩
```

```
        'дb''b''ob''b''чb''b''eb''b''pb''b''нb''b''иb''b''eb'' b''yb''b''зb''b''лb''b''ыb'' ↩
         b''вb'' b''cb''b''ob''b''ob''b''тb''b''вb''b''eb''b''тb''b''cb''b''тb''b''вb''b' ↩
        'иb''b''иb'' b''cb'' b''тb''b''eb''b''кb''b''yb''b''щb''b''иb''b''мb''b''иb''
    * b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иb''b''яb''b''мb''b''иb'' b''гb''b' ↩
        'лb''b''ob''b''бb''b''ab''b''лb''b''ьb''b''нb''b''ыb''b''хb'' b''иb''b''нb''b''дb'' ↩
        b''eb''b''кb''b''cb''b''ob''b''вb'' SG*-b''кb''b''лb''b''ab''b''cb''b''cb''b''ab''
    *
    * @param aNode - b''yb''b''зb''b''eb''b''лb'' b''вb''b''eb''b''pb''b''хb''b''нb''b' ↩
        'eb''b''гb''b''ob'' b''yb''b''pb''b''ob''b''вb''b''нb''b''яb''
    */
    SGLIB_API void RenameNodes( SGNODE* aNode );

    /**
    * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' DestroyNode
    * b''yb''b''дb''b''ab''b''лb''b''яb''b''eb''b''тb'' b''пb''b''eb''b''pb''b''eb''b' ↩
        'дb''b''ab''b''нb''b''нb''b''ыb''b''йb'' b''yb''b''зb''b''eb''b''лb'' SG*-b''кb''b' ↩
        'лb''b''ab''b''cb''b''cb''b''ab''. b''Эb''b''тb''b''ab'' b''фb''b''yb''b''нb''b' ↩
        'кb''b''цb''b''иb''b''яb'' b''пb''b''ob''b''зb''b''вb''b''ob''b''лb''b''яb''b''eb'' ↩
        b''тb'' b''бb''b''eb''b''зb''b''ob''b''пb''b''ab''b''cb''b''нb''b''ob''
    * b''yb''b''дb''b''ab''b''лb''b''яb''b''тb''b''ьb'' SG*-b''yb''b''зb''b''лb''b''ыb'', ↩
        b''нb''b''eb'' b''пb''b''pb''b''иb''b''бb''b''eb''b''гb''b''ab''b''яb'' b''кb'' b' ↩
        'cb''b''вb''b''яb''b''зb''b''ыb''b''вb''b''ab''b''нb''b''иb''b''юb'' b''cb'' b' ↩
        'cb''b''ob''b''ob''b''тb''b''вb''b''eb''b''тb''b''cb''b''тb''b''вb''b''yb''b''юb''b ↩
        ''щb''b''eb''b''йb''
    * IFSG*-b''ob''b''бb''b''ob''b''лb''b''ob''b''чb''b''кb''b''ob''b''йb''.
    */
    SGLIB_API void DestroyNode( SGNODE* aNode );

    // b''Пb''b''Pb''b''Иb''b''Mb''b''Eb''b''Чb''b''Ab''b''Нb''b''Иb''b''Eb'': b''cb''b' ↩
        'лb''b''eb''b''дb''b''yb''b''юb''b''щb''b''иb''b''eb'' b''фb''b''yb''b''нb''b''кb''b ↩
        ''цb''b''иb''b''иb'' b''ob''b''бb''b''лb''b''eb''b''гb''b''чb''b''ab''b''юb''b''тb'' ↩
         b''cb''b''ob''b''зb''b''дb''b''ab''b''нb''b''иb''b''eb'' b''иb'' b''yb''b''дb''b' ↩
        'ab''b''лb''b''eb''b''нb''b''иb''b''eb'' b''cb''b''тb''b''pb''b''yb''b''кb''b''тb''b ↩
        ''yb''b''pb''
    // b''дb''b''ab''b''нb''b''нb''b''ыb''b''хb'' b''дb''b''лb''b''яb'' b''pb''b''eb''b' ↩
        'нb''b''дb''b''eb''b''pb''b''иb''b''нb''b''гb''b''ab''

    /**
    * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' GetModel
    * b''cb''b''ob''b''зb''b''дb''b''ab''b''ёb''b''тb'' b''пb''b''pb''b''eb''b''дb''b' ↩
        'cb''b''тb''b''ab''b''вb''b''лb''b''eb''b''нb''b''иb''b''eb'' S3DMODEL b''дb''b' ↩
        'лb''b''яb'' aNode (b''чb''b''иb''b''cb''b''тb''b''ыb''b''eb'' b''дb''b''ab''b' ↩
        'нb''b''нb''b''ыb''b''eb'', b''бb''b''eb''b''зb'' b''пb''b''pb''b''eb''b''ob''b' ↩
        'бb''b''ab''b''pb''b''зb''b''ob''b''вb''b''ab''b''нb''b''иb''b''йb'')
    *
    * @param aNode - b''yb''b''зb''b''eb''b''лb'', b''кb''b''ob''b''тb''b''ob''b''pb''b' ↩
        'ыb''b''йb'' b''нb''b''yb''b''жb''b''нb''b''ob'' b''пb''b''pb''b''eb''b''ob''b' ↩
        'бb''b''pb''b''ab''b''зb''b''ob''b''вb''b''ab''b''тb''b''ьb'' b''вb'' b''пb''b' ↩
```

```
        'pb''b''eb''b''дb''b''cb''b''тb''b''ab''b''вb''b''лb''b''eb''b''нb''b''иb''b''eb''  ↩
        S3DMODEL
     * @return - b''вb''b''ob''b''зb''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''тb'' b' ↩
        'пb''b''pb''b''eb''b''дb''b''cb''b''тb''b''ab''b''вb''b''лb''b''eb''b''нb''b''иb''b ↩
        ''eb'' S3DMODEL b''вb'' b''cb''b''лb''b''yb''b''чb''b''ab''b''eb'' b''yb''b''cb''b' ↩
        'пb''b''eb''b''xb''b''ab'', b''иb''b''нb''b''ab''b''чb''b''eb'' - NULL
     */
    SGLIB_API S3DMODEL* GetModel( SCENEGRAPH* aNode );

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' Destroy3DModel
     * b''ob''b''cb''b''вb''b''ob''b''бb''b''ob''b''жb''b''дb''b''ab''b''eb''b''тb'' b' ↩
        'пb''b''ab''b''мb''b''яb''b''тb''b''ьb'', b''зb''b''ab''b''нb''b''иb''b''мb''b' ↩
        'ab''b''eb''b''мb''b''yb''b''юb'' b''cb''b''тb''b''pb''b''yb''b''кb''b''тb''b''yb'' ↩
        b''pb''b''ob''b''йb'' S3DMODEL b''иb'' b''cb''b''cb''b''ыb''b''лb''b''ab''b''eb''b' ↩
        'тb'' b''yb''b''кb''b''ab''b''зb''b''ab''b''тb''b''eb''b''лb''b''ьb''
     * b''cb''b''тb''b''pb''b''yb''b''кb''b''тb''b''yb''b''pb''b''ыb'' b''нb''b''ab'' NULL
     */
    SGLIB_API void Destroy3DModel( S3DMODEL** aModel );

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' Free3DModel
     * b''ob''b''cb''b''вb''b''ob''b''бb''b''ob''b''жb''b''дb''b''ab''b''eb''b''тb'' b' ↩
        'пb''b''ab''b''мb''b''яb''b''тb''b''ьb'', b''зb''b''ab''b''нb''b''иb''b''мb''b' ↩
        'ab''b''eb''b''мb''b''yb''b''юb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''мb''b''иb'' ↩
         b''cb''b''тb''b''pb''b''yb''b''кb''b''тb''b''yb''b''pb''b''ыb'' S3DMODEL
     */
    SGLIB_API void Free3DModel( S3DMODEL& aModel );

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' Free3DMesh
     * b''ob''b''cb''b''вb''b''ob''b''бb''b''ob''b''жb''b''дb''b''ab''b''eb''b''тb'' b' ↩
        'пb''b''ab''b''мb''b''яb''b''тb''b''ьb'', b''зb''b''ab''b''нb''b''иb''b''мb''b' ↩
        'ab''b''eb''b''мb''b''yb''b''юb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''мb''b''иb'' ↩
         b''cb''b''тb''b''pb''b''yb''b''кb''b''тb''b''yb''b''pb''b''ыb'' SMESH
     */
    SGLIB_API void Free3DMesh( SMESH& aMesh );

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' New3DModel
     * b''cb''b''ob''b''зb''b''дb''b''ab''b''ёb''b''тb'' b''иb'' b''иb''b''нb''b''иb''b' ↩
        'цb''b''иb''b''ab''b''лb''b''иb''b''зb''b''иb''b''pb''b''yb''b''eb''b''тb'' b''cb'' ↩
        b''тb''b''pb''b''yb''b''кb''b''тb''b''yb''b''pb''b''yb'' S3DMODEL
     */
    SGLIB_API S3DMODEL* New3DModel( void );

    /**
     * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иb''b''яb'' Init3DMaterial
```

```
     * b''ив''b''нв''b''ив''b''цв''b''ив''b''ав''b''лв''b''ив''b''зв''b''ив''b''рв''b''ув'' ↩
         b''ев''b''тв'' b''св''b''тв''b''рв''b''ув''b''кв''b''тв''b''ув''b''рв''b''ув''  ↩
         SMATERIAL
     */
    SGLIB_API void Init3DMaterial( SMATERIAL& aMat );

    /**
     * b''Фв''b''ув''b''нв''b''кв''b''цв''b''ив''b''яв'' Init3DMesh
     * b''св''b''ов''b''зв''b''дв''b''ав''b''ёв''b''тв'' b''ив'' b''ив''b''нв''b''ив''b' ↩
         'цв''b''ив''b''ав''b''лв''b''ив''b''зв''b''ив''b''рв''b''ув''b''ев''b''тв'' b''св'' ↩
         b''тв''b''рв''b''ув''b''кв''b''тв''b''ув''b''рв''b''ув'' SMESH
     */
    SGLIB_API void Init3DMesh( SMESH& aMesh );
};
```

Примеры реального использования API графа сцены можно посмотреть в примере 3D-плагина DEMO2 и в исходных кодах KiCad — 3D-плагины для работы с файлами в форматах VRML1, VRML2 и X3D.