



## **Плагины KiCad**

**25 апреля 2019 г.**

---

# Содержание

<b>1</b>	<b>Введение в систему плагинов KiCad</b>	<b>2</b>
1.1	Классы плагинов . . . . .	2
1.1.1	Класс плагинов PLUGIN_3D . . . . .	3
<b>2</b>	<b>Примеры: Класс 3D-плагинов</b>	<b>6</b>
2.1	Простой 3D-плагин . . . . .	6
2.2	Сложный 3D-плагин . . . . .	14
<b>3</b>	<b>Интерфейс программирования приложений (API)</b>	<b>24</b>
3.1	API класса плагинов . . . . .	25
3.1.1	API: базовый класс плагинов KiCad . . . . .	25
3.1.2	API: класс 3D-плагинов . . . . .	26
3.2	API калсса графа сцены . . . . .	28

## *Система плагинов KiCad*

### **Авторские права**

Авторские права © 2016 на данный документ принадлежит его разработчикам (соавторам), перечисленным ниже. Документ можно распространять и/или изменять в соответствии с правилами лицензии GNU General Public License (<http://www.gnu.org/licenses/gpl.html>), версии 3 или более поздней, или лицензии типа Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0/>), версии 3.0 или более поздней.

Все торговые знаки этого руководства принадлежат его владельцам.

### **Соавторы**

Cirilo Bernardo

### **Перевод**

Барановский Константин <[baranovskiykonstantin@gmail.com](mailto:baranovskiykonstantin@gmail.com)>, 2016-2019

### **Обратная связь**

Оставить свои комментарии или замечания можно на следующих ресурсах:

- О документации KiCad: <https://github.com/KiCad/kicad-doc/issues>
- О программном обеспечении KiCad: <https://bugs.launchpad.net/kicad>
- О переводе программного обеспечения KiCad: <https://github.com/KiCad/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 основных 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 большого 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''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''иб''b''mb''b' ←
'яb'' b''kb''b''lb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''lb''b''ab''b''gb''b''иб''b' ←
'нb''b''ab'' b''vb'' b''vb''b''иб''b''дb''b''eb'' b''cb''b''tb''b''pb''b''ob''b''kb''b' ←
'иб'' UTF-8 */
char const* GetKicadPluginClass( void );

/* b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''иб''b''нb''b' ←
'фb''b''ob''b''pb''b''mb''b''ab''b''цb''b''иб''b''юb'' b''об'' b''vb''b''eb''b''pb''b' ←
'cb''b''иб''b''иб'' API b''kb''b''lb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''lb''b''ab'' ←
b''gb''b''иб''b''нb''b''ab'' */
void GetClassVersion( unsigned char* Major, unsigned char* Minor,
    unsigned char* Patch, unsigned char* Revision );

/*
b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''иб''b''cb''b' ←
'tb''b''иб''b''нb''b''yb'', b''eb''b''cb''b''lb''b''иб'' b''pb''b''eb''b''ab''b''пb'' ←
b''иб''b''zb''b''ob''b''vb''b''ab''b''нb''b''нb''b''ab''b''яb'' b''пb''b''pb''b''ob'' ←
b''vb''b''eb''b''pb''b''kb''b''ab'' b''vb''b''eb''b''pb''b''cb''b''иб''b''йb'' b' ←
'vb'' b''пb''b''пb''b''ab''b''gb''b''иб''b''нb''b''eb''
b''об''b''пb''b''pb''b''eb''b''дb''b''eb''b''пb''b''иб''b''пb''b''ab'', b''чb''b''tb''b' ←
'об'' b''yb''b''kb''b''ab''b''zb''b''ab''b''нb''b''нb''b''ыb''b''йb'' API b''kb''b' ←
'пb''b''ab''b''cb''b''cb''b''ab'' -- b''cb''b''ob''b''vb''b''mb''b''eb''b''cb''b' ←
'tb''b''иб''b''mb''.

*/
bool CheckClassVersion( unsigned char Major,
    unsigned char Minor, unsigned char Patch, unsigned char Revision );

/* b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''иб''b''mb''b' ←
'яb'' b''дb''b''ab''b''нb''b''нb''b''ob''b''gb''b''ob'' b''пb''b''пb''b''ab''b''gb''b' ←
'иб''b''нb''b''ab'', b''нb''b''ab''b''пb''b''pb''b''иб''b''mb''b''eb''b''pb'', " ←
PLUGIN_3D_VRML" */
const char* GetKicadPluginName( void );

/* b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''иб''b''нb''b' ←
'фb''b''ob''b''pb''b''mb''b''ab''b''цb''b''иб''b''юb'' b''об'' b''vb''b''eb''b''pb''b' ←
'cb''b''иб''b''иб'' b''дb''b''ab''b''нb''b''нb''b''ob''b''gb''b''ob'' b''пb''b''пb''b' ←
'ab''b''gb''b''иб''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-плагинов, а также указано несколько функций, которые пользователь не должен изменять. Следующие функции не должны реализовываться пользователем:

```

/* b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''шb''b''ab''b''eb''b''tb'' b''иб''b''mb''b' ←
'яb'' b''kb''b''lb''b''ab''b''cb''b''cb''b''ab'' b''пb''b''lb''b''ab''b''gb''b''иб''b' ←
'нb''b''ab'' -- "PLUGIN_3D" */
char const* GetKicadPluginClass( void );

/* b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''шb''b''ab''b''eb''b''tb'' b''иб''b''нb''b' ←
'fb''b''ob''b''pb''b''mb''b''ab''b''цb''b''иб''b''юb'' b''об'' b''vb''b''eb''b''pb''b' ←
'cb''b''иб''b''иб'' API b''kb''b''lb''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''yb''b''пb''b''ob''b''lb''b''нb''b''яb''b''eb''b''tb'' b''об''b''бb''b''yb''b' ←
'чb''b''нb''b''yb''b''юb'' b''пb''b''pb''b''ob''b''vb''b''eb''b''pb''b''kb''b''yb'' b ←
''vb''b''eb''b''pb''b''cb''b''иб''b''иб'', b''pb''b''eb''b''ab''b''lb''b''иб''b''zb'' ←
b''ob''b''vb''b''ab''b''нb''b''нb''b''yb''b''юb'' b''pb''b''ab''b''zb''b''pb''b''ab'' ←
b''бb''b''ob''b''tb''b''чb''b''иб''b''kb''b''ab''b''mb''b''иб''
b''zb''b''ab''b''gb''b''pb''b''yb''b''zb''b''чb''b''иб''b''kb''b''ab'' b''kb''b''lb''b' ←
'ab''b''cb''b''cb''b''ab'' b''пb''b''lb''b''ab''b''gb''b''иб''b''нb''b''ob''b''vb'' ←
PLUGIN_3D, b''иб'' b''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''шb''b''ab''b''eb''b' ←
'tb'' b''иб''b''cb''b''tb''b''иб''b''нb''b''yb'', b''eb''b''cb''b''lb''b''иб''
b''пb''b''pb''b''ob''b''vb''b''eb''b''pb''b''kb''b''ab'' b''yb''b''cb''b''пb''b''eb''b' ←
'шb''b''нb''b''ob'' b''пb''b''pb''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''zb''b''vb''b''pb''b''ab''b''шb''b''ab''b''eb''b''tb'' b''kb''b''ob''b' ←
'lb''b''иб''b''чb''b''eb''b''cb''b''tb''b''vb''b''ob'' b''cb''b''tb''b''pb''b''ob''b' ←
'kb'' b''cb'' b''pb''b''ab''b''cb''b''шb''b''иб''b''pb''b''eb''b''нb''b''иб''b''яb''b' ←
'mb''b''иб'', b''kb''b''ob''b''tb''b''ob''b''pb''b''yb''b''eb'' b''пb''b''ob''b' ←
'дb''b''дb''b''eb''b''pb''b''жb''b''иб''b''vb''b''ab''b''юb''b''tb''b''cb''b''яb''
b''пb''b''lb''b''ab''b''gb''b''иб''b''нb''b''ob''b''mb''
*/
int GetNExtensions( void );

/*
b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''шb''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''yb''b''юb'' b''cb''b''tb''b''pb''b' ←
'ob''b''kb''b''yb'' b''cb'' b''pb''b''ab''b''cb''b''шb''b''иб''b''pb''b''eb''b''нb''b' ←
'иб''b''eb''b''mb''; b''дb''b''ob''b''cb''b''tb''b''yb''b''пb''b''нb''b''yb'' b' ←
'zb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иб''b''яb'' b''ob''b''tb'' 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''шb''b''ab''b''eb''b''tb'' b''ob''b''бb''b' ←
        'шb''b''eb''b''eb'' b''kb''b''ob''b''lb''b''иб''b''чb''b''eb''b''cb''b''tb''b''vb''b' ←
        'ob'' b''fb''b''иб''b''lb''b''ьb''b''tb''b''pb''b''ob''b''vb'' b''tb''b''иб''b''пb''b' ←
        ''ob''b''vb'' b''fb''b''ab''b''йb''b''lb''b''ob''b''vb'', b''kb''b''ob''b''tb''b' ←
        'ob''b''pb''b''ьb''b''eb''
    b''пb''b''ob''b''дb''b''дb''b''eb''b''pb''b''жb''b''иб''b''vb''b''ab''b''юb''b''tb''b' ←
        'cb''b''яb'' b''пb''b''lb''b''ab''b''gb''b''иб''b''нb''b''ob''b''mb''
*/
int GetNFilters( void );

/*
    b''Bb''b''ob''b''zb''b''vb''b''pb''b''ab''b''шb''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''ьb''b''йb'' b''fb''b''иб''b''пb''b' ←
        'ьb''b''tb''b''pb'' b''tb''b''иб''b''пb''b''ob''b''vb'' b''fb''b''ab''b''йb''b''пb''b' ←
        ''ob''b''vb''; b''дb''b''ob''b''cb''b''tb''b''yb''b''пb''b''нb''b''ьb'' b''zb''b' ←
        'нb''b''ab''b''чb''b''eb''b''нb''b''иб''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''шb''b''ab''b''eb''b''tb'' b''иб''b''cb''b' ←
        'tb''b''иб''b''нb''b''yb'', b''eb''b''cb''b''lb''b''иб'' b''пb''b''пb''b''ab''b' ←
        'gb''b''иб''b''нb'' b''mb''b''ob''b''жb''b''eb''b''tb'' b''ob''b''tb''b''ob''b''бb'' ←
        b''pb''b''ab''b''zb''b''иб''b''tb''b''ьb'' b''дb''b''ab''b''нb''b''нb''b''ьb''b' ←
        'йb'' b''tb''b''иб''b''пb'' 3D-b''mb''b''ob''b''дb''b''eb''b''пb''b''иб''.
    b''Bb'' b''нb''b''eb''b''kb''b''ob''b''tb''b''ob''b''pb''b''ьb''b''xb'' b''cb''b''пb''b' ←
        ''yb''b''чb''b''ab''b''яb''b''xb'', b''пb''b''пb''b''ab''b''gb''b''иб''b''нb'' b' ←
        'нb''b''eb'' b''mb''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''vb''b''иб''b''tb''b''ьb'' b''vb''b''иб''b''zb''b''yb''b' ←
        'ab''b''пb''b''ьb''b''нb''b''yb''b''юb'' b''mb''b''ob''b''дb''b''eb''b''пb''b''ьb''
    b''иб'' b''дb''b''ob''b''пb''b''жb''b''eb''b''нb'' b''vb''b''eb''b''pb''b''нb''b''yb''b' ←
        ''tb''b''ьb'' b''пb''b''ob''b''жb''b''ьb''.
*/
bool CanRender( void );

/*
    b''Зb''b''ab''b''gb''b''pb''b''yb''b''zb''b''иб''b''tb''b''ьb'' b''yb''b''kb''b''ab''b' ←
        'zb''b''ab''b''нb''b''нb''b''yb''b''юb'' b''mb''b''ob''b''дb''b''eb''b''пb''b''ьb'' b' ←
        ''иб'' b''vb''b''eb''b''pb''b''нb''b''yb''b''tb''b''ьb'' b''yb''b''kb''b''ab''b''zb'' ←
        b''ab''b''tb''b''eb''b''пb''b''ьb'' b''нb''b''ab'' b''дb''b''ab''b''нb''b''нb''b' ←
        'ьb''b''eb'' b''eb''b''ёb'' b''vb''b''иб''b''zb''b''yb''b''ab''b''пb''b''ьb''b''нb''b' ←
        ''ob''b''gb''b''ob''
    b''пb''b''pb''b''eb''b''дb''b''cb''b''tb''b''ab''b''vb''b''пb''b''eb''b''нb''b''иб''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''pb''b''ob''b''pb''b''yb''b''tb''b''kb''b''ab'' b''ib''b''zb''b''vb''b''lb''b''eb'' ←
    b''чb''b''ьb'' b''иб''b''нb''b''fb''b''ob''b''pb''b''mb''b''ab''b''цb''b''иб''b'' ←
    'yb'' b''ob'' b''vb''b''eb''b''pb''b''cb''b''иб''b''иб'' b''иб''b''zb'' b''fb''b' ←
    'ab''b''йb''b''lb''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''иб''b''зб''b''вб''b''лб''b''еб''b''чб''b''еб''b''нб''b''иб''b''еб'' b''cb''b' ←
'tb''b''pb''b''об''b''кб''b''иб'' "#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''лб''b''pb''b''иб''b''вб''b''еб''b''cb''b''tb''b''иб'' b''нб''b''еб''b''yb''b' ←
'кб''b''аб''b''зб''b''аб''b''нб''b''нб''b''ыб''b''еб'' b''чб''b''аб''b''tb''b' ←
'cb''b''иб'' b''вб''b''еб''b''pb''b''cb''b''иб''b''иб'' b''кб'' b''нб''b''yb''b' ←
'лб''b''юб''

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''иб''b''mb''b''яb'' b''пb''b''pb''b' ←
'ob''b''eb''b''kb''b''tb''b''ab''
project( PLUGIN_DEMO )

# b''пb''b''pb''b''ob''b''vb''b''eb''b''pb''b''иб''b''tb''b''ьb'', b''yb''b''cb''b''tb''b' ←
'ab''b''nb''b''ob''b''vb''b''lb''b''eb''b''nb''b''ab'' b''lb''b''иб'' b''nb''b''yb''b' ←
'jb''b''nb''b''ab''b''яb'' b''vb''b''eb''b''pb''b''cb''b''иб''b''яb'' CMake b''cb''b' ←
'ob'' b''vb''b''cb''b''eb''b''mb''b''иб'' b''nb''b''yb''b''jb''b''nb''b''ьb''b''mb''b' ←
'иб'' b''cb''b''vb''b''ob''b''йb''b''cb''b''tb''b''vb''b''ab''b''mb''b''иб''
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''tb''b' ←
'ob'' b''дb''b''lb''b''яb'' b''пb''b''ob''b''иб''b''cb''b''kb''b''ab'' b''cb''b''kb''b' ←
'pb''b''иб''b''пb''b''tb''b''ab'' FindKICAD
set( CMAKE_MODULE_PATH ${PROJECT_SOURCE_DIR}/CMakeModules )

```

```
# b''пб''b''об''b''пб''b''ыб''b''тб''b''кб''b''аб'' b''нб''b''аб''b''йб''b''тб''b''иб'' b' ←
'yb''b''сб''b''тб''b''аб''b''нб''b''об''b''вб''b''лб''b''еб''b''нб''b''нб''b''ыб''b' ←
'еб'' b''зб''b''аб''b''гб''b''об''b''лб''b''об''b''вб''b''об''b''чб''b''нб''b''ыб''b' ←
'еб'' b''фб''b''аб''b''йб''b''лб''b''ыб'' b''иб'' b''бб''b''иб''b''бб''b''лб''b''иб''b' ←
'об''b''тб''b''еб''b''кб''b''yb'' KiCad

# b''иб'' b''об''b''пб''b''рб''b''еб''b''дб''b''еб''b''лб''b''нб''b''тб''b''ьб'' b''пб''b' ←
'еб''b''рб''b''еб''b''мб''b''еб''b''нб''b''нб''b''ыб''b''еб''':

# KICAD_INCLUDE_DIR
# KICAD_LIBRARY
find_package( KICAD 1.0 REQUIRED )

# b''дб''b''об''b''бб''b''аб''b''вб''b''иб''b''тб''b''ьб'' b''кб''b''аб''b''тб''b''аб''b' ←
'лб''b''об''b''гб'' b''зб''b''аб''b''гб''b''об''b''лб''b''об''b''вб''b''об''b''чб''b' ←
'нб''b''ыб''b''хб'' b''фб''b''аб''b''йб''b''лб''b''об''b''вб'' kicad b''кб'' b''пб''b' ←
'yb''b''тб''b''яб''b''мб'' b''пб''b''об''b''иб''b''сб''b''кб''b''аб'' b''кб''b''об''b' ←
'мб''b''пб''b''иб''b''лб''b''яб''b''тб''b''об''b''рб''b''аб''

include_directories( ${KICAD_INCLUDE_DIR}/kicad )

# b''сб''b''об''b''зб''b''дб''b''аб''b''тб''b''ьб'' b''пб''b''лб''b''аб''b''гб''b''иб''b' ←
'нб'' b''сб'' b''иб''b''мб''b''еб''b''нб''b''еб''b''мб'' 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''вб'' b''зб''b''аб''b''гб''b''об''b''лб''b''об''b''вб''b''об''b''чб''b''нб''b''об''b' ←
'мб'' b''фб''b''аб''b''йб''b''лб''b''еб'' 3d_plugin.h b''об''b''бб''b''ьб''b''яб''b' ←
'вб''b''лб''b''еб''b''нб''b''ыб'' b''фб''b''yb''b''нб''b''кб''b''цб''b''иб''b''иб'', b' ←
'об''b''бб''b''яб''b''зб''b''аб''b''тб''b''еб''b''лб''b''ьб''b''нб''b''ыб''b''еб'' b' ←
'дб''b''лб''b''яб''

// 3D-b''пб''b''лб''b''аб''b''гб''b''иб''b''нб''b''об''b''вб''
#include "plugins/3d/3d_plugin.h"

// b''yb''b''кб''b''аб''b''жб''b''иб''b''тб''b''еб'' b''иб''b''нб''b''фб''b''об''b''рб''b' ←
'мб''b''аб''b''цб''b''иб''b''юб'' b''об'' b''вб''b''еб''b''рб''b''сб''b''иб''b''иб'' b' ←
```

```

    'дб''б''аб''б''нб''б''нб''б''об''б''гб''б''об'' б''пб''б''лб''б''аб''б''гб''б''иб''б' ←
    'нб''б''аб''; б''нб''б''еб'' б''пб''б''уб''б''тб''б''аб''б''йб''б''тб''б''еб'' б''эб''б' ←
    'тб''б''об'' б''сб''
// б''вб''б''еб''б''пб''б''сб''б''иб''б''еб''б''йб'' б''кб''б''лб''б''аб''б''сб''б''сб''б' ←
    'аб'' б''пб''б''лб''б''аб''б''гб''б''иб''б''нб''б''аб'', б''кб''б''об''б''тб''б''об''б' ←
    'пб''б''аб''б''яб'' б''уб''б''кб''б''аб''б''зб''б''аб''б''нб''б''аб'' б''вб'' 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

// б''пб''б''еб''б''аб''б''лб''б''иб''б''зб''б''уб''б''йб''б''тб''б''еб'' б''фб''б''уб''б' ←
    'нб''б''кб''б''цб''б''иб''б''юб'', б''кб''б''об''б''тб''б''об''б''пб''б''аб''б''яб'' б' ←
    'пб''б''пб''б''еб''б''дб''б''об''б''сб''б''тб''б''аб''б''вб''б''лб''б''яб''б''еб''б' ←
    'тб'' б''пб''б''об''б''лб''б''ьб''б''зб''б''об''б''вб''б''аб''б''тб''б''еб''б''лб''б' ←
    'яб''б''мб'' б''иб''б''мб''б''яб'' б''пб''б''лб''б''аб''б''гб''б''иб''б''нб''б''аб''
const char* GetKicadPluginName( void )
{
    return "PLUGIN_3D_DEMO1";
}

// б''пб''б''еб''б''аб''б''лб''б''иб''б''зб''б''уб''б''йб''б''тб''б''еб'' б''фб''б''уб''б' ←
    'нб''б''кб''б''цб''б''иб''б''юб'', б''кб''б''об''б''тб''б''об''б''пб''б''аб''б''яб'' б' ←
    'пб''б''пб''б''еб''б''дб''б''об''б''сб''б''тб''б''аб''б''вб''б''лб''б''яб''б''еб''б' ←
    'тб'' б''пб''б''об''б''лб''б''ьб''б''зб''б''об''б''вб''б''аб''б''тб''б''еб''б''лб''б' ←
    'яб''б''мб'' б''вб''б''еб''б''пб''б''сб''б''иб''б''юб'' б''пб''б''лб''б''аб''б''гб''б' ←
    'иб''б''нб''б''аб''
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;
}

// б''кб''б''об''б''лб''б''иб''б''чб''б''еб''б''сб''б''тб''б''вб''б''об'' б''пб''б''об''б' ←
    'дб''б''дб''б''еб''б''пб''б''жб''б''иб''б''вб''б''аб''б''еб''б''мб''б''ыб''б''xb'' б' ←

```

```

'pb''b''ab''b''cb''b''шб''b''иб''b''pb''b''eb''b''нб''b''иб''b''йб''; b''нб''b''ab'' b' ←
'cb''b''иб''b''cb''b''тб''b''eb''b''мб''b''ab''b''xb'' *NIX b''pb''b''ab''b''cb''b''шб'' ←
b''иб''b''pb''b''eb''b''нб''b''иб''b''яб''
// b''yb''b''kb''b''ab''b''зб''b''ыб''b''вб''b''ab''b''юб''b''тб''b''cb''b''яб'' b''дб''b' ←
'вб''b''ab''b''жб''b''дб''b''ыб'' - b''об''b''дб''b''нб''b''об'' b''вб'' b''нб''b''иб''b' ←
''жб''b''нб''b''eb''b''мб'' b''pb''b''eb''b''гб''b''иб''b''cb''b''тб''b''pb''b''eb'', b' ←
'вб''b''тб''b''об''b''pb''b''об''b''eb'' - b''вб'' b''вб''b''eb''b''pb''b''xb''b''нб''b' ←
'eb''b''мб''
#ifdef _WIN32
#define NEXTS 7
#else
#define NEXTS 14
#endif

// b''kb''b''об''b''лб''b''иб''b''чб''b''eb''b''cb''b''тб''b''вб''b''об'' b''пб''b''об''b' ←
'дб''b''дб''b''eb''b''pb''b''жб''b''иб''b''вб''b''ab''b''eb''b''мб''b''ыб''b''xb'' b' ←
'фб''b''иб''b''лб''b''ыб''b''тб''b''pb''b''об''b''вб'' b''тб''b''иб''b''пб''b''об''b' ←
'вб'' b''фб''b''ab''b''йб''b''лб''b''об''b''вб''
#define NFILS 5

// b''об''b''пб''b''pb''b''eb''b''дб''b''eb''b''лб''b''иб''b''тб''b''eb'' b''cb''b''тб''b' ←
'pb''b''об''b''kb''b''иб'' b''cb'' b''pb''b''ab''b''cb''b''шб''b''иб''b''pb''b''eb''b' ←
'нб''b''иб''b''яб''b''мб''b''иб'' b''иб'' b''фб''b''иб''b''лб''b''ыб''b''тб''b''pb''b' ←
'аб''b''мб''b''иб'', b''kb''b''об''b''тб''b''об''b''pb''b''ыб''b''eb'' b''пб''b''об''b' ←
'дб''b''дб''b''eb''b''pb''b''жб''b''иб''b''вб''b''ab''b''eb''b''тб''
// b''дб''b''ab''b''нб''b''нб''b''ыб''b''йб'' b''пб''b''лб''b''ab''b''гб''b''иб''b''нб''
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;*.STEP";
#endif

// b''ob''b''пb''b''pb''b''eb''b''дb''b''eb''b''лb''b''иб''b''тb''b''eb'' b''cb''b''тb''b' ←
'pb''b''yb''b''kb''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''об''b''гb''b''об'' b''дb''b''об''b''cb''b''тb''b''yb''b''пb''b' ←
'ab'' b''kb'' b''дb''b''ab''b''нb''b''нb''b''ыb''b''мb''
// b''вb'' b''вb''b''иб''b''дb''b''eb'' b''cb''b''пb''b''иб''b''cb''b''kb''b''об''b''вb'' b ←
'cb''b''тb''b''pb''b''об''b''kb'' b''pb''b''ab''b''cb''b''шb''b''иб''b''pb''b''eb''b' ←
'нb''b''иб''b''йb'' b''иб'' b''фb''b''иб''b''лb''b''ьb''b''тb''b''pb''b''об''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;

#ifdef _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;

```

```

// б''вб''б''об''б''зб''б''вб''б''рб''б''аб''б''шб''б''аб''б''еб''б''тб'' б''кб''б''об''б' ←
'лб''б''иб''б''чб''б''еб''б''сб''б''тб''б''вб''б''об'' б''рб''б''аб''б''сб''б''шб''б' ←
'иб''б''рб''б''еб''б''нб''б''иб''б''йб'', б''пб''б''об''б''дб''б''дб''б''еб''б''рб''б' ←
'жб''б''иб''б''вб''б''аб''б''еб''б''мб''б''ыб''б''хб'' б''эб''б''тб''б''иб''б''мб'' б' ←
'пб''б''лб''б''аб''б''рб''б''иб''б''нб''б''об''б''мб''

int GetNExtensions( void )
{
    return NEXTS;
}

// б''вб''б''об''б''зб''б''вб''б''рб''б''аб''б''шб''б''аб''б''еб''б''тб'' б''сб''б''тб''б' ←
'рб''б''об''б''кб''б''yb'' б''рб''б''аб''б''сб''б''шб''б''иб''б''рб''б''еб''б''нб''б' ←
'иб''б''яб'' б''пб''б''об'' б''yb''б''кб''б''аб''б''зб''б''аб''б''нб''б''нб''б''об''б' ←
'мб''б''yb'' б''иб''б''нб''б''дб''б''еб''б''кб''б''сб''б''yb''

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

    return file_data.extensions[aIndex];
}

// б''вб''б''об''б''зб''б''вб''б''рб''б''аб''б''шб''б''аб''б''еб''б''тб'' б''кб''б''об''б' ←
'лб''б''иб''б''чб''б''еб''б''сб''б''тб''б''вб''б''об'' б''сб''б''тб''б''рб''б''об''б' ←
'кб'' б''фб''б''иб''б''лб''б''ьб''б''тб''б''рб''б''об''б''вб'', б''пб''б''рб''б''еб''б' ←
'дб''б''об''б''сб''б''тб''б''аб''б''вб''б''лб''б''яб''б''еб''б''мб''б''ыб''б''хб'' б' ←
'эб''б''тб''б''иб''б''мб'' б''пб''б''лб''б''аб''б''рб''б''иб''б''нб''б''об''б''мб''

int GetNFilters( void )
{
    return NFILS;
}

// б''вб''б''об''б''зб''б''вб''б''рб''б''аб''б''шб''б''аб''б''еб''б''тб'' б''сб''б''тб''б' ←
'рб''б''об''б''кб''б''yb'' б''фб''б''иб''б''лб''б''ьб''б''тб''б''рб''б''аб'' б''пб''б' ←
'об'' б''yb''б''кб''б''аб''б''зб''б''аб''б''нб''б''нб''б''об''б''мб''б''yb'' б''иб''б' ←
'нб''б''дб''б''еб''б''кб''б''сб''б''yb''

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

    return file_data.filters[aIndex];
}

// б''вб''б''об''б''зб''б''вб''б''рб''б''аб''б''шб''б''аб''б''еб''б''тб'' б''лб''б''об''б' ←
'жб''б''ьб'', б''еб''б''сб''б''лб''б''иб'' б''пб''б''лб''б''аб''б''рб''б''иб''б''нб'' б' ←

```



```

    'нб''б''еб'' б''пб''б''об''б''дб''б''гб''б''об''б''тб''б''об''б''вб''б''иб''б''лб'' б' ←
    'дб''б''аб''б''нб''б''нб''б''ыб''б''еб'' б''вб''б''иб''б''зб''б''уб''б''аб''б''лб''б' ←
    'иб''б''зб''б''аб''б''цб''б''иб''б''иб''
bool CanRender( void )
{
    return false;
}

// б''вб''б''об''б''зб''б''вб''б''рб''б''аб''б''щб''б''аб''б''еб''б''тб'' NULL б''пб''б' ←
'об''б''кб''б''аб'' б''пб''б''лб''б''аб''б''гб''б''иб''б''нб'' б''нб''б''еб'' б''пб''б' ←
'об''б''дб''б''гб''б''об''б''тб''б''об''б''вб''б''иб''б''тб'' б''дб''б''аб''б''нб''б' ←
'нб''б''ыб''б''еб'' б''вб''б''иб''б''зб''б''уб''б''аб''б''лб''б''иб''б''зб''б''аб''б' ←
'цб''б''иб''б''иб''
SCENEGRAPH* Load( char const* aFileName )
{
    // б''эб''б''тб''б''об''б''тб'' б''пб''б''рб''б''иб''б''мб''б''иб''б''тб''б''иб''б' ←
    'вб''б''нб''б''ыб''б''йб'' б''пб''б''лб''б''аб''б''гб''б''иб''б''нб'' б''нб''б''еб'' ←
    б''пб''б''об''б''дб''б''дб''б''еб''б''рб''б''жб''б''иб''б''вб''б''аб''б''еб''б' ←
    'тб'' б''рб''б''еб''б''нб''б''дб''б''еб''б''рб''б''иб''б''нб''б''гб'' б''нб''б''иб'' ←
    б''кб''б''аб''б''кб''б''иб''б''хб'' б''мб''б''об''б''дб''б''еб''б''лб''б''еб''б' ←
    'йб''
    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}
# б''эб''б''кб''б''цб''б''пб''б''об''б''рб''б''тб''б''иб''б''рб''б''уб''б''йб''б''тб''б' ←
'еб'' KICAD_ROOT_DIR, б''еб''б''цб''б''лб''б''иб'' б''пб''б''об''б''нб''б''аб''б''дб''б' ←
'об''б''бб''б''иб''б''тб''б''цб''б''яб''
mkdir build && cd build
cmake .. && make

```

Плагин будет построен, но не установлен. Можно скопировать его в каталог, 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 завершён и файлы скриптов `CMakeLists.txt` и `Find-KICAD.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 трёхмерных 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''иб''b''яb'' b''иб''b''зб'' b' ←
'kb''b''лb''b''ab''b''cb''b''cb''b''ab'' 3D-b''пb''b''лb''b''ab''b''гb''b''иб''b''нb''b' ←
'ob''b''вb''

#include "plugins/3d/3d_plugin.h"
// b''иб''b''нb''b''тb''b''eb''b''pb''b''fb''b''eb''b''йb''b''cb'' b''дб''b''лb''b''яb'' b' ←
'pb''b''ab''b''бb''b''ob''b''тb''b''ыb'' b''cb'' b''бb''b''иб''b''бb''b''лb''b''иб''b' ←
'ob''b''тb''b''eb''b''kb''b''ob''b''йb'' b''гb''b''pb''b''ab''b''fb''b''ab'' b''cb''b' ←
'цб''b''eb''b''нb''b''ыb'' b''иб''b''зб'' KiCad

#include "plugins/3dapi/ifsg_all.h"

// b''иб''b''нb''b''fb''b''ob''b''pb''b''мb''b''ab''b''цб''b''иб''b''яb'' b''об'' b''вb''b' ←
'eb''b''pb''b''cb''b''иб''b''иб'' b''дб''b''лb''b''яb'' b''дб''b''ab''b''нb''b''нb''b' ←
'ob''b''гb''b''ob'' b''пb''b''лb''b''ab''b''гb''b''иб''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
```

```

// б''пб''б''рб''б''еб''б''дб''б''об''б''сб''б''тб''б''аб''б''вб''б''лб''б''яб''б''еб''б' ←
    'тб'' б''иб''б''мб''б''яб'' б''дб''б''аб''б''нб''б''нб''б''об''б''гб''б''об'' б''пб''б' ←
    'лб''б''аб''б''гб''б''иб''б''нб''б''аб''
const char* GetKicadPluginName( void )
{
    return "PLUGIN_3D_DEMO2";
}

// б''пб''б''рб''б''еб''б''дб''б''об''б''сб''б''тб''б''аб''б''вб''б''лб''б''яб''б''еб''б' ←
    'тб'' б''вб''б''еб''б''рб''б''сб''б''иб''б''юб'' б''дб''б''аб''б''нб''б''нб''б''об''б' ←
    'гб''б''об'' б''пб''б''лб''б''аб''б''гб''б''иб''б''нб''б''аб''
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;
}

// б''кб''б''об''б''лб''б''иб''б''чб''б''еб''б''сб''б''тб''б''вб''б''об'' б''пб''б''об''б' ←
    'дб''б''дб''б''еб''б''рб''б''жб''б''иб''б''вб''б''аб''б''еб''б''мб''б''ыб''б''хб'' б' ←
    'рб''б''аб''б''сб''б''шб''б''иб''б''рб''б''еб''б''нб''б''иб''б''йб''
#ifdef _WIN32
#define NEXTS 1
#else
#define NEXTS 2
#endif

// б''кб''б''об''б''лб''б''иб''б''чб''б''еб''б''сб''б''тб''б''вб''б''об'' б''пб''б''об''б' ←
    'дб''б''дб''б''еб''б''рб''б''жб''б''иб''б''вб''б''аб''б''еб''б''мб''б''ыб''б''хб'' б' ←
    'фб''б''иб''б''лб''б''ьб''б''тб''б''рб''б''об''б''вб''
#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;

#ifdef _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''nb''b''ёb''b''tb'' b''иб''b''cb''b''tb''b''иб''b''nb''b''yb'', b ←
'kb''b''ob''b''rb''b''db''b''ab'' b''пb''b''лb''b''ab''b''rb''b''иб''b''nb'' b''cb''b' ←
'mb''b''ob''b''jb''b''eb''b''tb'' b''пb''b''pb''b''eb''b''db''b''ob''b''cb''b''tb''b' ←
'ab''b''вb''b''иб''b''tb''b''ьb'' b''db''b''ab''b''nb''b''nb''b''yb''b''eb'' b''вb''b' ←
'иб''b''zb''b''yb''b''ab''b''лb''b''иб''b''zb''b''ab''b''цb''b''иб''b''иб''
bool CanRender( void )
{
    return true;
}

// b''cb''b''ob''b''zb''b''db''b''ab''b''nb''b''иб''b''eb'' b''db''b''ab''b''nb''b''nb''b' ←
'yb''b''xb'' b''вb''b''иб''b''zb''b''yb''b''ab''b''лb''b''иб''b''zb''b''ab''b''цb''b' ←
'иб''b''иб''
SCENEGRAPH* Load( char const* aFileName )
{
    // b''db''b''лb''b''яb'' b''эb''b''tb''b''ob''b''rb''b''ob'' b''пb''b''pb''b''иб''b' ←
'mb''b''eb''b''pb''b''ab'' b''бb''b''yb''b''db''b''eb''b''tb'' b''cb''b''ob''b''zb'' ←
b''db''b''ab''b''nb'' b''tb''b''eb''b''tb''b''pb''b''ab''b''эb''b''db''b''pb'' (tx1) ←
, b''cb''b''ob''b''db''b''eb''b''pb''b''jb''b''ab''b''щb''b''иб''b''йb''b''cb''b' ←
'яb'' b''вb'' b''rb''b''pb''b''ab''b''fb''b''eb''
    // b''cb''b''цb''b''eb''b''nb''b''yb'' SCENEGRAPH (VRML Transform) b''иб'' b''cb''b' ←
'ob''b''cb''b''tb''b''ob''b''яb''b''щb''b''иб''b''йb'' b''иб''b''zb'' b''чb''b''eb'' ←
b''tb''b''yb''b''pb''b''eb''b''xb'' b''ob''b''бb''b''ьb''b''eb''b''kb''b''tb''b' ←
'ob''b''вb''
    // b''rb''b''pb''b''ab''b''nb''b''eb''b''йb'' SGSHAPE (VRML Shape) b''db''b''лb''b' ←
'яb'' b''kb''b''ab''b''jb''b''db''b''ob''b''йb'' b''иб''b''zb'' b''eb''b''rb''b' ←
'ob'' b''cb''b''tb''b''ob''b''pb''b''ob''b''nb''. b''Kb''b''ab''b''jb''b''db''b' ←
'ob''b''йb'' b''rb''b''pb''b''ab''b''nb''b''иб''
    // b''пb''b''pb''b''иб''b''cb''b''вb''b''ab''b''иб''b''вb''b''ab''b''eb''b''tb''b''cb'' ←
b''яb'' b''цb''b''вb''b''eb''b''tb'' (SGAPPEARANCE) b''иб'' SGFACESET (VRML Geometry ←
->indexedFaceSet).
    // b''Kb''b''ab''b''jb''b''db''b''yb''b''йb'' SGFACESET b''cb''b''вb''b''яb''b''zb''b' ←
'yb''b''вb''b''ab''b''eb''b''tb''b''cb''b''яb'' b''cb''b''ob'' b''cb''b''пb''b''иб'' ←
b''cb''b''kb''b''ob''b''mb'' b''вb''b''eb''b''pb''b''шb''b''иб''b''nb'' (SGCOORDS), ←
b''cb''b''пb''b''иб''b''cb''b''kb''b''ob''b''mb''
    // b''вb''b''eb''b''kb''b''tb''b''ob''b''pb''b''ob''b''вb'' (SGNORMALS) b''иб'' b''иб'' ←
b''nb''b''db''b''eb''b''kb''b''cb''b''ab''b''mb''b''иб'' b''kb''b''ob''b''ob''b' ←
'pb''b''db''b''иб''b''nb''b''ab''b''tb'' (SGCOORDINDEX). b''Ob''b''db''b''nb''b' ←
'ab'' b''rb''b''pb''b''ab''b''nb''b''ьb''

```

```

// b''иб''b''cb''b''пб''b''об''b''лб''b''ьб''b''зб''b''yb''b''eb''b''tb''b''cb''b''яб'' ←
    b''дб''b''лб''b''яб'' b''пб''b''рб''b''eb''b''дб''b''cb''b''tb''b''аб''b''вб''b'' ←
    'лб''b''eb''b''нб''b''иб''b''яб'' b''об''b''дб''b''нб''b''об''b''йб'' b''иб''b''зб'' ←
    b''cb''b''tb''b''об''b''рб''b''об''b''нб'', b''tb''b''аб''b''кб'' b''чб''b''tb''b'' ←
    'об'' b''мб''b''об''b''жб''b''нб''b''об''
// b''иб''b''cb''b''пб''b''об''b''лб''b''ьб''b''зб''b''об''b''вб''b''аб''b''tb''b''ьб'' ←
    b''вб''b''eb''b''кб''b''tb''b''об''b''рб''b''ьб'' b''вб''b''eb''b''рб''b''шб''b'' ←
    'иб''b''нб''-b''гб''b''рб''b''аб''b''нб''b''eb''b''йб'' (per-vertex-per-face normals ←
    ).
//
// b''Эб''b''tb''b''об''b''tb'' b''tb''b''eb''b''tb''b''рб''b''аб''b''эб''b''дб''b'' ←
    'рб'' b''яб''b''вб''b''лб''b''яб''b''eb''b''tb''b''cb''b''яб'' b''дб''b''об''b''чб'' ←
    b''eb''b''рб''b''нб''b''иб''b''мб'', b''пб''b''об'' b''об''b''tb''b''нб''b''об''b'' ←
    'шб''b''eb''b''нб''b''иб''b''юб'' b''кб'' b''эб''b''лб''b''eb''b''мб''b''eb''b''нб'' ←
    b''tb''b''yb'' b''вб''b''eb''b''рб''b''кб''b''нб''b''eb''b''гб''b''об''
// b''yb''b''рб''b''об''b''вб''b''нб''b''яб'' SCENEGRAPH (tx0), b''вб'' b''кб''b''об''b'' ←
    'tb''b''об''b''рб''b''об''b''мб'' b''cb''b''об''b''дб''b''eb''b''рб''b''жб''b''иб'' ←
    b''tb''b''cb''b''яб'' b''вб''b''tb''b''об''b''рб''b''об''b''йб'' b''дб''b''об''b'' ←
    'чб''b''eb''b''рб''b''нб''b''иб''b''йб''
// SCENEGRAPH (tx2), b''кб''b''об''b''tb''b''об''b''рб''b''ьб''b''йб'' b''вб'' b''cb''b'' ←
    'вб''b''об''b''юб'' b''об''b''чб''b''eb''b''рб''b''eb''b''дб''b''ьб'', b''яб''b'' ←
    'вб''b''лб''b''яб''b''eb''b''tb''b''cb''b''яб'' b''рб''b''eb''b''зб''b''yb''b''лб''b'' ←
    'ьб''b''tb''b''аб''b''tb''b''об''b''мб''
// b''пб''b''рб''b''eb''b''об''b''бб''b''рб''b''аб''b''зб''b''об''b''вб''b''аб''b''нб'' ←
    b''иб''b''яб'' b''tb''b''eb''b''tb''b''рб''b''аб''b''эб''b''дб''b''рб''b''аб'' tx1 ( ←
    b''пб''b''об''b''вб''b''об''b''рб''b''об''b''tb'' + b''cb''b''мб''b''eb''b''шб''b'' ←
    'eb''b''нб''b''иб''b''eb''). b''Эб''b''tb''b''иб''b''мб'' b''бб''b''yb''b''дб''b'' ←
    'eb''b''tb''
// b''пб''b''об''b''кб''b''аб''b''зб''b''аб''b''нб''b''об'' b''кб''b''аб''b''кб'' b'' ←
    'пб''b''об''b''вб''b''tb''b''об''b''рб''b''нб''b''об'' b''иб''b''cb''b''пб''b''об''b'' ←
    'лб''b''ьб''b''зб''b''об''b''вб''b''аб''b''tb''b''ьб'' b''эб''b''лб''b''eb''b''мб'' ←
    b''eb''b''нб''b''tb''b''ьб'' b''вб'' b''иб''b''eb''b''рб''b''аб''b''рб''b''кб''b'' ←
    'иб''b''иб'' b''гб''b''рб''b''аб''b''фб''b''аб'' b''cb''b''цб''b''eb''b''нб''b'' ←
    'ьб''.

// b''об''b''бб''b''ьб''b''яб''b''вб''b''лб''b''eb''b''нб''b''иб''b''eb'' b''вб''b'' ←
    'eb''b''рб''b''шб''b''иб''b''нб'' b''tb''b''eb''b''tb''b''рб''b''аб''b''эб''b''дб''b'' ←
    'рб''b''аб''
// 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 );

```

```

// б'сб'б'об'б'зб'б'дб'б'аб'б'нб'б'иб'б'еб' б'об'б'бб'б'ьб'б' ←
'еб'б'кб'б'тб'б'аб' б'пб'б'рб'б'еб'б'об'б'бб'б'рб'б'аб'б'зб'б' ←
'об'б'вб'б'аб'б'нб'б'иб'б'яб' б'вб'б'еб'б'рб'б'хб'б'нб'б'еб' ←
б'гб'б'об' б'уб'б'рб'б'об'б'вб'б'нб'б'яб'; б'об'б'нб' б'бб'б' ←
'уб'б'дб'б'еб'б'тб' б'сб'б'об'б'дб'б'еб'б'рб'б'жб'б'аб'б'тб'б' ←
'ьб'

// б'вб'б'сб'б'еб' б'об'б'сб'б'тб'б'аб'б'пб'б'ьб'б'нб'б'ьб'б' ←
'еб' б'об'б'бб'б'ьб'б'еб'б'кб'б'тб'б'ьб' б'гб'б'рб'б'аб'б'фб' ←
б'об'б'вб' б'сб'б'цб'б'еб'б'нб'б'ьб'; б'об'б'бб'б'ьб'б'еб'б' ←
'кб'б'тб' б'пб'б'рб'б'еб'б'об'б'бб'б'рб'б'аб'б'зб'б'об'б'вб'б' ←
'аб'б'нб'б'иб'б'яб' б'мб'б'об'б'жб'б'еб'б'тб'

// б'сб'б'об'б'дб'б'еб'б'рб'б'жб'б'аб'б'тб'б'ьб' б'дб'б'об'б' ←
'чб'б'еб'б'рб'б'нб'б'иб'б'еб' б'об'б'бб'б'ьб'б'еб'б'кб'б'тб'б' ←
'ьб' б'пб'б'рб'б'еб'б'об'б'бб'б'рб'б'аб'б'зб'б'об'б'вб'б'аб' ←
б'нб'б'иб'б'яб' б'иб'б'лб'б'иб' б'гб'б'рб'б'аб'б'нб'б'иб'

IFSG_TRANSFORM* tx0 = new IFSG_TRANSFORM( true );

// б'сб'б'об'б'зб'б'дб'б'аб'б'тб'б'ьб' б'об'б'бб'б'ьб'б'еб'б' ←
'кб'б'тб' б'пб'б'рб'б'еб'б'об'б'бб'б'рб'б'аб'б'зб'б'об'б'вб'б' ←
'аб'б'нб'б'иб'б'яб', б'вб' б'кб'б'об'б'тб'б'об'б'рб'б'об'б' ←
'мб' б'бб'б'уб'б'дб'б'уб'б'тб' б'хб'б'рб'б'аб'б'нб'б'иб'б'тб' ←
б'сб'б'яб' б'гб'б'рб'б'аб'б'нб'б'иб'

IFSG_TRANSFORM* tx1 = new IFSG_TRANSFORM( tx0->GetRawPtr() );

// б'дб'б'об'б'бб'б'аб'б'вб'б'иб'б'тб'б'ьб' б'гб'б'рб'б'аб'б' ←
'нб'б'ьб', б'кб'б'об'б'тб'б'об'б'рб'б'аб'б'яб' б'бб'б'уб'б' ←
'дб'б'еб'б'тб' б'сб'б'лб'б'уб'б'жб'б'нб'б'тб'б'ьб' б'об'б'дб' ←
б'нб'б'об'б'йб' б'иб'б'зб' б'сб'б'тб'б'об'б'рб'б'об'б'нб' б' ←
'тб'б'еб'б'тб'б'рб'б'аб'б'эб'б'дб'б'рб'б'аб';

// б'гб'б'рб'б'аб'б'нб'б'иб' б'сб'б'об'б'сб'б'тб'б'об'б'яб'б' ←
'тб' б'иб'б'зб' б'нб'б'аб'б'бб'б'об'б'рб'б'аб' б'сб'б'тб'б' ←
'об'б'рб'б'об'б'нб' б'иб' б'аб'б'тб'б'рб'б'иб'б'бб'б'уб'б'тб' ←
б'об'б'вб' б'вб'б'нб'б'еб'б'шб'б'нб'б'еб'б'гб'б'об' б'вб'б' ←
'иб'б'дб'б'аб' (appearances)

IFSG_SHAPE* shape = new IFSG_SHAPE( *tx1 );

// б'дб'б'об'б'бб'б'аб'б'вб'б'иб'б'тб'б'ьб' б'нб'б'аб'б'бб'б' ←
'об'б'рб' б'сб'б'тб'б'об'б'рб'б'об'б'нб'; б'об'б'нб' б'сб'б' ←
'об'б'сб'б'тб'б'об'б'иб'б'тб' б'иб'б'зб' б'сб'б'пб'б'иб'б'сб' ←
б'кб'б'аб' б'кб'б'об'б'об'б'рб'б'дб'б'иб'б'нб'б'аб'б'тб', б' ←
'иб'б'нб'б'дб'б'еб'б'кб'б'сб'б'об'б'вб'

// б'кб'б'об'б'об'б'рб'б'дб'б'иб'б'нб'б'аб'б'тб', б'сб'б'пб'б' ←
'иб'б'сб'б'кб'б'аб' б'вб'б'еб'б'рб'б'шб'б'иб'б'нб', б'иб'б' ←
'нб'б'дб'б'еб'б'кб'б'сб'б'об'б'вб' б'вб'б'еб'б'рб'б'шб'б'иб'б' ←
'нб' б'иб' б'мб'б'об'б'жб'б'еб'б'тб' б'сб'б'об'б'дб'б'еб'б' ←
'рб'б'жб'б'аб'б'тб'б'ьб' б'сб'б'пб'б'иб'б'сб'б'об'б'кб'

```

```

// б''цб''б''вб''б''еб''б''тб''б''об''б''вб'' б''иб'' б''иб''б''хб'' б''иб''б''нб''б' ←
'дб''б''еб''б''кб''б''сб''б''ыб''.

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] );

// б''иб''б''нб''б''дб''б''еб''б''кб''б''сб''б''ыб'' б''кб''б''об''б''об''б''рб''б' ←
'дб''б''иб''б''нб''б''аб''б''тб'' - б''пб''б''рб''б''иб''б''мб''б''еб''б''чб''б' ←
'аб''б''нб''б''иб''б''еб'': б''иб''б''сб''б''пб''б''об''б''лб''б''ьб''б''зб''б''уб'' ←
б''юб''б''тб''б''сб''б''яб'' б''тб''б''рб''б''еб''б''уб''б''рб''б''об''б''лб''б' ←
'ьб''б''нб''б''иб''б''кб''б''иб'';
// б''нб''б''аб'' б''пб''б''рб''б''аб''б''кб''б''тб''б''иб''б''кб''б''еб'', б''пб''б' ←
'лб''б''аб''б''рб''б''иб''б''нб''б''ыб'', б''кб''б''об''б''тб''б''об''б''рб''б''ыб'' ←
б''еб'' б''нб''б''еб'' б''мб''б''об''б''рб''б''уб''б''тб'' б''об''б''пб''б''рб''б' ←
'еб''б''дб''б''еб''б''лб''б''иб''б''тб''б''ьб'' б''кб''б''аб''б''кб''б''аб''б''яб''
// б''иб''б''зб'' б''сб''б''тб''б''об''б''рб''б''об''б''нб'' б''тб''б''рб''б''еб''б' ←
'уб''б''рб''б''об''б''лб''б''ьб''б''нб''б''иб''б''кб''б''аб'' б''бб''б''уб''б''дб''б' ←
'еб''б''тб'' б''вб''б''иб''б''дб''б''нб''б''аб'', б''иб''б''сб''б''пб''б''об''б' ←
'лб''б''ьб''б''зб''б''уб''б''юб''б''тб'' б''пб''б''об'' б''дб''б''вб''б''еб'' б' ←
'тб''б''об''б''чб''б''кб''б''иб''
// б''дб''б''лб''б''яб'' б''кб''б''аб''б''жб''б''дб''б''об''б''рб''б''об'' б''тб''б' ←
'рб''б''еб''б''уб''б''рб''б''об''б''лб''б''ьб''б''нб''б''иб''б''кб''б''аб''
IFSG_COORDINDEX* coordIdx = new IFSG_COORDINDEX( *face );
coordIdx->AddIndex( 0 );
coordIdx->AddIndex( 1 );
coordIdx->AddIndex( 2 );

// б''об''б''пб''б''рб''б''еб''б''дб''б''еб''б''лб''б''иб''б''тб''б''ьб'' б''аб''б' ←
'тб''б''рб''б''иб''б''бб''б''уб''б''тб''б''ыб''; б''аб''б''тб''б''рб''б''иб''б''бб'' ←
б''уб''б''тб''б''ыб'' б''пб''б''рб''б''иб''б''нб''б''аб''б''дб''б''лб''б''еб''б' ←
'жб''б''аб''б''тб'' б''рб''б''рб''б''аб''б''нб''б''иб''

// б''пб''б''уб''б''рб''б''пб''б''уб''б''рб''б''нб''б''ыб''б''йб''
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 );

// б''вб''б''еб''б''кб''б''тб''б''об''б''рб''б''ыб''
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 );

//
// б''Гб''б''pb''б''ab''б''нб''б''ьб'' 2
// б''Пб''б''pb''б''иб''б''мб''б''еб''б''чб''б''аб''б''нб''б''иб''б''еб'': б''об''б' ←
    'бб''б''ёб''б''pb''б''тб''б''кб''б''аб'' IFSG* б''иб''б''сб''б''пб''б''об''б''лб''б' ←
    'ьб''б''зб''б''yb''б''еб''б''тб''б''сб''б''яб'' б''пб''б''об''б''вб''б''тб''б''об''б' ←
    'pb''б''нб''б''об'' б''дб''б''лб''б''яб'' б''сб''б''об''б''зб''б''дб''б''аб''б' ←
    'нб''б''иб''б''яб'' б''иб''
// б''yb''б''пб''б''pb''б''аб''б''вб''б''лб''б''еб''б''нб''б''иб''б''яб'' б''сб''б' ←
    'тб''б''pb''б''yb''б''кб''б''тб''б''yb''б''pb''б''аб''б''мб''б''иб'' б''дб''б''аб''б' ←
    'нб''б''нб''б''ьб''б''xb''.
//
shape->NewNode( *tx1 );
face->NewNode( *shape );
coordIdx->NewNode( *face );
cp->NewNode( *face );
np->NewNode( *face );

// б''вб''б''еб''б''pb''б''шб''б''иб''б''нб''б''ьб''
cp->AddCoord( vert[0] );
cp->AddCoord( vert[2] );
cp->AddCoord( vert[3] );

// б''иб''б''нб''б''дб''б''еб''б''кб''б''сб''б''ьб''
coordIdx->AddIndex( 0 );
coordIdx->AddIndex( 1 );
coordIdx->AddIndex( 2 );

// б''вб''б''еб''б''кб''б''тб''б''об''б''pb''б''ьб''
nval = S3D::CalcTriNorm( vert[0], vert[2], vert[3] );
np->AddNormal( nval );
np->AddNormal( nval );
np->AddNormal( nval );
// б''цб''б''вб''б''еб''б''тб'' (б''кб''б''pb''б''аб''б''сб''б''нб''б''ьб''б''йб'')
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 );

//
// б''Гб''б''pb''б''аб''б''нб''б''ьб'' 3
//
shape->NewNode( *tx1 );
face->NewNode( *shape );
coordIdx->NewNode( *face );
cp->NewNode( *face );

```

```

np->NewNode( *face );

// б''вб''б''еб''б''пб''б''шб''б''иб''б''нб''б''ыб''
cp->AddCoord( vert[1] );
cp->AddCoord( vert[3] );
cp->AddCoord( vert[2] );

// б''иб''б''нб''б''дб''б''еб''б''кб''б''сб''б''ыб''
coordIdx->AddIndex( 0 );
coordIdx->AddIndex( 1 );
coordIdx->AddIndex( 2 );

// б''вб''б''еб''б''кб''б''тб''б''об''б''пб''б''ыб''
nval = S3D::CalcTriNorm( vert[1], vert[3], vert[2] );
np->AddNormal( nval );
np->AddNormal( nval );
np->AddNormal( nval );

// б''цб''б''вб''б''еб''б''тб'' (б''зб''б''еб''б''лб''б''ёб''б''нб''б''ыб''б''йб'')
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 );

//
// б''Гб''б''пб''б''аб''б''нб''б''ыб'' 4
//
shape->NewNode( *tx1 );
face->NewNode( *shape );
coordIdx->NewNode( *face );
cp->NewNode( *face );
np->NewNode( *face );

// б''вб''б''еб''б''пб''б''шб''б''иб''б''нб''б''ыб''
cp->AddCoord( vert[0] );
cp->AddCoord( vert[1] );
cp->AddCoord( vert[2] );

// б''иб''б''нб''б''дб''б''еб''б''кб''б''сб''б''ыб''
coordIdx->AddIndex( 0 );
coordIdx->AddIndex( 1 );
coordIdx->AddIndex( 2 );

// б''вб''б''еб''б''кб''б''тб''б''об''б''пб''б''ыб''
nval = S3D::CalcTriNorm( vert[0], vert[1], vert[2] );
np->AddNormal( nval );
np->AddNormal( nval );

```

```

np->AddNormal( nval );

// b''цб''b''вб''b''eb''b''тб'' (b''cb''b''иб''b''нб''b''иб''b''йб'')
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''об''b''зб''b''дб''b''аб''b''тб''b''ьб'' b''кб''b''об''b''пб''b''иб''b' ←
'юб'' b''вб''b''cb''b''eb''b''гб''b''об'' b''тб''b''eb''b''тб''b''пб''b''аб''b''эб'' ←
b''дб''b''пб''b''аб'', b''cb''b''мб''b''eb''b''cb''b''тб''b''иб''b''тб''b''ьб'' b' ←
'eb''b''ёб'' b''пб''b''об'' b''об''b''cb''b''иб'' z b''нб''b''аб'' 2 (Z+2) b''иб''
// b''нб''b''об''b''вб''b''eb''b''пб''b''нб''b''yb''b''тб''b''ьб'' b''нб''b''аб'' 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''дб''b''аб''b''пб''b''иб''b''тб''b''ьб'' b''пб''b''eb''b''пб''b''eb''b' ←
'мб''b''eb''b''нб''b''нб''b''ьб''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 защищённая (`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''tb''b''kb''b''pb''b''yb''b''tb''b''yb'' b''pb''b''lb''b''ab''b''gb''b''ib''b' ←
  'nb'', b''yb''b''kb''b''ab''b''zb''b''ab''b''nb''b''nb''b''yb''b''yb'' b''vb'' b''vb''b' ←
  'ib''b''db''b''eb'' b''pb''b''ob''b''lb''b''nb''b''ob''b''gb''b''ob'' b''pb''b''yb''b' ←
  'tb''b''ib'' "aFullFileName" */
bool Open( const wxString& aFullFileName );

/* b''3b''b''ab''b''kb''b''pb''b''yb''b''tb''b''yb'', b''ob''b''tb''b''kb''b''pb''b''yb''b' ←
  'tb''b''yb''b''yb'' b''vb'' b''db''b''ab''b''nb''b''nb''b''yb''b''yb'' b''mb''b''ob''b' ←
  'mb''b''eb''b''nb''b''tb'', b''pb''b''lb''b''ab''b''gb''b''ib''b''nb''.*/
void Close( void );

/* b''Пб''b''об''b''лб''b''yb''b''чб''b''иб''b''tb''b''yb'' b''vb''b''eb''b''pb''b''cb''b' ←
  'иб''b''юб'' API b''kb''b''лб''b''ab''b''cb''b''cb''b''ab'' b''пб''b''лб''b''ab''b''gb'' ←
  b''иб''b''nb''b''ob''b''vb'', b''pb''b''eb''b''ab''b''лб''b''иб''b''zb''b''ob''b''vb''b' ←
  'ab''b''nb''b''nb''b''yb''b''юб'' b''db''b''ab''b''nb''b''nb''b''yb''b''mb'' b''zb''b' ←
  'ab''b''gb''b''pb''b''yb''b''zb''b''чб''b''иб''b''kb''b''ob''b''mb'' */
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''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щб''b''ab''b''eb''b''tb'' b''иб''b''mb''b' ←
  'яб'' b''kb''b''лб''b''ab''b''cb''b''cb''b''ab'' b''пб''b''лб''b''ab''b''gb''b''иб''b' ←
  'nb''b''ob''b''vb'' b''иб''b''лб''b''иб'' NULL, b''eb''b''cb''b''лб''b''иб'' b''пб''b' ←
  'лб''b''ab''b''gb''b''иб''b''nb'' b''nb''b''eb'' b''zb''b''ab''b''gb''b''pb''b''yb''b' ←
  'жб''b''eb''b''nb'' */
char const* GetKicadPluginClass( void );

/* b''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щб''b''ab''b''eb''b''tb'' b''лб''b''об''b' ←
  'жб''b''ьб'', b''eb''b''cb''b''лб''b''иб'' b''пб''b''лб''b''ab''b''gb''b''иб''b''nb'' b' ←
  'nb''b''eb'' b''zb''b''ab''b''gb''b''pb''b''yb''b''жб''b''eb''b''nb'' */
bool GetClassVersion( 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''щб''b''ab''b''eb''b''tb'' b''лб''b''об''b' ←
  'жб''b''ьб'', b''eb''b''cb''b''лб''b''иб'' b''vb''b''eb''b''pb''b''cb''b''иб''b''яб'' b' ←
  'kb''b''лб''b''ab''b''cb''b''cb''b''ab'' b''nb''b''eb'' b''cb''b''ob''b''vb''b''mb''b' ←
  'eb''b''cb''b''tb''b''иб''b''mb''b''ab'' b''иб''b''лб''b''иб'' b''пб''b''лб''b''ab''b' ←
```

```

    'gb''b''ib''b''nb'' b''nb''b''eb'' b''zb''b''ab''b''gb''b''pb''b''yb''b''jb''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''щb''b''ab''b''eb''b''tb'' b''ib''b''mb''b' ←
    'яb'' b''пb''b''лb''b''ab''b''gb''b''ib''b''nb''b''ab'' b''ib''b''лb''b''ib'' NULL, b' ←
    'eb''b''cb''b''лb''b''ib'' b''пb''b''лb''b''ab''b''gb''b''ib''b''nb'' b''nb''b''eb'' b' ←
    'zb''b''ab''b''gb''b''pb''b''yb''b''jb''b''eb''b''nb'' */
const char* GetKicadPluginName( void );

/*
    b''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''лb''b''ob''b' ←
    'jb''b''ьb'', b''eb''b''cb''b''лb''b''ib'' b''пb''b''лb''b''ab''b''gb''b''ib''b''nb'' ←
    b''nb''b''eb'' b''zb''b''ab''b''gb''b''pb''b''yb''b''jb''b''eb''b''nb'', b''vb'' b' ←
    'пb''b''pb''b''ob''b''tb''b''ib''b''vb''b''nb''b''ob''b''mb'' b''cb''b''лb''b''yb''b' ←
    'чb''b''ab''b''eb''
    b''vb'' b''пb''b''ab''b''pb''b''ab''b''mb''b''eb''b''tb''b''pb''b''ab''b''xb'' b''бb''b' ←
    'yb''b''дb''b''eb''b''tb'' b''cb''b''ob''b''дb''b''eb''b''pb''b''jb''b''ab''b''tb''b' ←
    'ьb''b''cb''b''яb'' b''pb''b''eb''b''zb''b''yb''b''лb''b''ьb''b''tb''b''ab''b''tb'' b' ←
    'fb''b''yb''b''nb''b''xb''b''цb''b''ib''b''ib'' GetPluginInfo()
*/
bool GetVersion( unsigned char* Major, unsigned char* Minor,
    unsigned char* Patch, unsigned char* Revision );

/*
    b''eb''b''cb''b''лb''b''ib'' b''пb''b''лb''b''ab''b''gb''b''ib''b''nb'' b''nb''b''eb'' b' ←
    ''zb''b''ab''b''gb''b''pb''b''yb''b''jb''b''eb''b''nb'', b''yb''b''cb''b''tb''b''ab'' ←
    b''nb''b''ab''b''vb''b''лb''b''nb''b''vb''b''ab''b''eb''b''tb'' b''zb''b''nb''b''ab'' ←
    b''чb''b''eb''b''nb''b''ib''b''eb'' b''пb''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''дb''b''eb'' b''пb''b''yb''b''cb''b''tb''b''ob''b''йb'' b''cb''b' ←
    ''tb''b''pb''b''ob''b''xb''b''ib''; b''vb'' b''пb''b''pb''b''ob''b''tb''b''ib''b' ←
    'vb''b''nb''b''ob''b''mb'' b''cb''b''лb''b''yb''b''чb''b''ab''b''eb'', b''zb''b''nb'' ←
    b''ab''b''чb''b''eb''b''nb''b''ib''b''eb'' b''ob''b''бb''b''pb''b''ab''b''zb''b''yb'' ←
    b''eb''b''tb''b''cb''b''яb'' b''cb''b''лb''b''eb''b''дb''b''yb''b''юb''b''щb''b''ib'' ←
    b''mb''
    b''ob''b''бb''b''pb''b''ab''b''zb''b''ob''b''mb'':
    [NAME]:[MAJOR].[MINOR].[PATCH].[REVISION]
    b''gb''b''дb''b''eb'':
    NAME = b''ib''b''mb''b''яb'', b''пb''b''ob''b''лb''b''yb''b''чb''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''цb'' ←
    b''ib''b''яb'' b''ob'' b''vb''b''eb''b''pb''b''cb''b''ib''b''ib'', b''пb''b''ob''b' ←
    'лb''b''yb''b''чb''b''eb''b''nb''b''nb''b''ab''b''яb'' 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 класса 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''pb''b''pb''b''eb''b''db''b''eb''b''pb''b''eb''b''nb''b''ib''b''eb'' b''ib''b' ←
'nb''b''fb''b''ob''b''pb''b''mb''b''ab''b''cb''b''ib''b''ib'' b''ob'' b''vb''b''eb''b ←
''pb''b''cb''b''ib''b''ib'' b''kb''b''pb''b''ab''b''cb''b''cb''b''ab'' b''rb''b''pb'' ←
b''ab''b''fb''b''ab'' b''cb''b''cb''b''eb''b''nb''b''yb''.
b''Bb''b''cb''b''eb'' b''pb''b''pb''b''ab''b''rb''b''ib''b''nb''b''yb'', b''ib''b''cb''b ←
''pb''b''ob''b''pb''b''yb''b''zb''b''yb''b''yb''b''yb''b''yb''b''yb''b''yb''b''yb'' ←
''ab''b''cb''b''cb'' b''rb''b''pb''b''ab''b''fb''b''ab'' b''cb''b''cb''b''eb''b''nb'' ←
b''yb'' b''db''b''ob''b''pb''b''jb''b''nb''b''yb'' b''vb''b''kb''b''pb''b''yb''b' ←
'cb''b''ab''b''tb''b''yb'' b''zb''b''tb''b''ob''b''tb''
b''zb''b''ab''b''rb''b''ob''b''pb''b''ob''b''vb''b''ob''b''cb''b''nb''b''yb''b''yb'' b' ←
'fb''b''ab''b''yb''b''pb'' b''ib'' b''pb''b''pb''b''ob''b''vb''b''eb''b''pb''b''yb''b ←
''tb''b''yb'' b''vb''b''eb''b''pb''b''cb''b''ib''b''yb'' b''kb''b''ab''b''jb''b''db'' ←
b''yb''b''yb'' b''pb''b''ab''b''zb'', b''ib''b''cb''b''pb''b''ob''b''pb''b''yb''b' ←
'zb''b''yb''b''yb'' b''pb''b''eb''b''zb''b''yb''b''pb''b''yb''b''tb''b''ab''b''tb''
b''fb''b''yb''b''nb''b''kb''b''cb''b''ib''b''ib'' S3D::GetLibVersion(), b''db''b''pb''b' ←
'yb'' b''pb''b''ob''b''db''b''tb''b''vb''b''eb''b''pb''b''jb''b''db''b''eb''b''nb''b' ←
'ib''b''yb'' b''cb''b''ob''b''vb''b''mb''b''eb''b''cb''b''tb''b''ib''b''mb''b''ob''b' ←
'cb''b''tb''b''ib''
*/

#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''pb''b''eb''b''db''b''eb''b''pb''b''eb''b''nb''b''ib''b''eb'' b''tb''b' ←
        'ib''b''pb''b''ob''b''vb'' b''db''b''pb''b''яb'' b''kb''b''pb''b''ab''b''cb''b''cb''b' ←
        ''ab'' b''gb''b''pb''b''ab''b''fb''b''ab'' b''cb''b''цb''b''eb''b''nb''b''yb''; b' ←
        'эb''b''tb''b''ib'' b''tb''b''ib''b''pb''b''yb''
    b''mb''b''ab''b''kb''b''cb''b''ib''b''mb''b''ab''b''pb''b''yb''b''nb''b''ob'' b''pb''b' ←
        'pb''b''ib''b''бb''b''pb''b''ib''b''жb''b''eb''b''nb''b''yb'' b''kb'' b''tb''b''ib''b' ←
        ''pb''b''ab''b''mb'' b''yb''b''эb''b''pb''b''ob''b''vb'' 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''tb''b''ab'' b''mb''b''ob''b''db''b''eb''b''pb''b''yb'' RGB-b''цb''b''vb''b' ←
        'eb''b''tb''b''ab'' b''ab''b''nb''b''ab''b''pb''b''ob''b''gb''b''ib''b''чb''b''nb''b' ←
        ''ab'' b''mb''b''ob''b''db''b''eb''b''pb''b''ib'' VRML2.0, b''gb''b''db''b''eb'' b' ←
        'kb''b''ab''b''жb''b''db''b''ob''b''mb''b''yb''
    b''цb''b''vb''b''eb''b''tb''b''yb'' b''pb''b''pb''b''ib''b''cb''b''vb''b''ab''b''ib''b' ←
        'vb''b''ab''b''eb''b''tb''b''cb''b''яb'' b''эb''b''nb''b''ab''b''чb''b''eb''b''nb''b' ←
        ''ib''b''eb'' b''vb'' b''db''b''ib''b''ab''b''pb''b''ab''b''эb''b''ob''b''nb''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''иб''b''мб''b''еб''b''еб''b''тб'' 3 b''сб''b''об''b''сб''b''тб''b''аб''b' ←
'вб''b''лб''b''яб''b''юб''b''щб''b''иб''b''еб'' (x,y,z), b''пб''b''об''b''дб''b' ←
'об''b''бб''b''нб''b''об'' b''тб''b''об''b''чб''b''кб''b''еб'', b''нб''b''об''
b''вб''b''еб''b''кб''b''тб''b''об''b''рб'' b''сб''b''об''b''дб''b''еб''b''рб''b''жб''b' ←
'иб''b''тб'' b''нб''b''об''b''рб''b''мб''b''аб''b''лб''b''иб''b''зб''b''об''b''вб''b' ←
''аб''b''нб''b''нб''b''ыб''b''еб'' b''зб''b''нб''b''аб''b''чб''b''еб''b''нб''b''иб'' ←
b''яб'' b''иб'' b''пб''b''рб''b''еб''b''дб''b''об''b''тб''b''вб''b''рб''b''аб''b' ←
'щб''b''аб''b''еб''b''тб''
b''иб''b''хб'' b''нб''b''еб''b''пб''b''об''b''сб''b''рб''b''еб''b''дб''b''сб''b''тб''b' ←
'вб''b''еб''b''нб''b''нб''b''об''b''еб'' b''иб''b''зб''b''мб''b''еб''b''нб''b''еб''b' ←
''нб''b''иб''b''еб''.
*/
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''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' Destroy
     * b''yb''b''дb''b''ab''b''лb''b''яb''b''eb''b''тb'' b''дb''b''ab''b''nb''b''nb''b' ←
       'yb''b''йb'' b''ob''b''бb''b''ъb''b''eb''b''kb''b''тb'' b''гb''b''pb''b''ab''b' ←
       'fb''b''ab'' b''cb''b''цb''b''eb''b''nb''b''yb''
     */
    void Destroy( void );

    /**
     * b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' Attach
     * b''cb''b''вb''b''яb''b''зb''b''ыb''b''вb''b''ab''b''eb''b''тb'' b''пb''b''ob''b' ←
       'лb''b''yb''b''чb''b''eb''b''nb''b''nb''b''yb''b''йb'' SGNODE* b''cb'' b''эb''b' ←
       'тb''b''иб''b''мb'' b''ob''b''бb''b''ъb''b''eb''b''kb''b''тb''b''ob''b''мb''
     */
    virtual bool Attach( SGNODE* aNode ) = 0;

    /**
     * b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' NewNode
     * b''cb''b''ob''b''зb''b''дb''b''ab''b''ёb''b''тb'' b''nb''b''ob''b''вb''b''yb''b' ←
       'йb'' b''yb''b''зb''b''eb''b''лb'' b''иб'' b''cb''b''вb''b''яb''b''зb''b''yb''b' ←
       'вb''b''ab''b''eb''b''тb'' b''eb''b''гb''b''ob'' b''cb'' b''эb''b''тb''b''иб''b' ←
       'мb'' b''ob''b''бb''b''ъb''b''eb''b''kb''b''тb''b''ob''b''мb''
     */
    virtual bool NewNode( SGNODE* aParent ) = 0;
    virtual bool NewNode( IFSG_NODE& aParent ) = 0;

    /**

```

```

* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' GetRawPtr()
* b''вb''b''об''b''зб''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''yb''b' ←
  'kb''b''ab''b''зб''b''ab''b''tb''b''eb''b''lb''b''ьb'' b''nb''b''eb''b''pb''b''об'' ←
  b''cb''b''pb''b''eb''b''дб''b''cb''b''tb''b''вb''b''eb''b''nb''b''nb''b''об'' b' ←
  'nb''b''ab'' SGNODE

*/

SGNODE* GetRawPtr( void );

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' GetNodeType
* b''вb''b''об''b''зб''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''tb''b' ←
  'иб''b''pb'' b''yb''b''зб''b''lb''b''ab'' b''дб''b''ab''b''nb''b''nb''b''об''b' ←
  'gb''b''об'' b''об''b''бб''b''ьb''b''eb''b''kb''b''tb''b''ab''

*/

S3D::SGTYPES GetNodeType( void ) const;

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' GetParent
* b''вb''b''об''b''зб''b''вb''b''pb''b''ab''b''щb''b''ab''b''eb''b''tb'' b''yb''b' ←
  'kb''b''ab''b''зб''b''ab''b''tb''b''eb''b''lb''b''ьb'' b''nb''b''ab'' b''pb''b' ←
  'об''b''дб''b''иб''b''tb''b''eb''b''lb''b''ьb''b''cb''b''kb''b''иб''b''йb'' SGNODE ←
  b''дб''b''lb''b''яb'' b''эб''b''tb''b''об''b''gb''b''об'' b''об''b''бб''b''ьb''b' ←
  'eb''b''kb''b''tb''b''ab''

* b''иб''b''lb''b''иб'' NULL, b''eb''b''cb''b''lb''b''иб'' b''об''b''бб''b''ьb''b' ←
  'eb''b''kb''b''tb'' b''nb''b''eb'' b''иб''b''mb''b''eb''b''eb''b''tb'' b''pb''b' ←
  'об''b''дб''b''иб''b''tb''b''eb''b''lb''b''eb''b''йb'' (b''tb''.b''eb''. b''яb''b' ←
  'вb''b''lb''b''яb''b''eb''b''tb''b''cb''b''яb''

* b''об''b''бб''b''ьb''b''eb''b''kb''b''tb''b''об''b''mb'' b''pb''b''pb''b''eb''b' ←
  'об''b''бб''b''pb''b''ab''b''зб''b''об''b''вb''b''ab''b''nb''b''иб''b''яb'' b''вb'' ←
  b''eb''b''pb''b''xb''b''nb''b''eb''b''gb''b''об'' b''yb''b''pb''b''об''b''вb''b' ←
  'nb''b''яb'') b''иб''b''lb''b''иб'', b''kb''b''об''b''gb''b''дб''b''ab'' b''дб''b' ←
  'ab''b''nb''b''nb''b''ьb''b''йb''

* b''об''b''бб''b''ьb''b''eb''b''kb''b''tb'' b''nb''b''eb'' b''cb''b''вb''b''яb''b' ←
  'зб''b''ab''b''nb'' b''cb'' SGNODE.

*/

SGNODE* GetParent( void ) const;

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' SetParent
* b''пb''b''pb''b''иб''b''cb''b''вb''b''ab''b''иб''b''вb''b''ab''b''eb''b''tb'' b' ←
  'pb''b''об''b''дб''b''иб''b''tb''b''eb''b''lb''b''ьb''b''cb''b''kb''b''иб''b''йb'' ←
  SGNODE b''дб''b''lb''b''яb'' b''дб''b''ab''b''nb''b''nb''b''об''b''gb''b''об'' b' ←
  'об''b''бб''b''ьb''b''eb''b''kb''b''tb''b''ab''.

*
* @param aParent [b''вb''b''xb''b''об''b''дб''b''яb''b''щb''b''иб''b''йb''] b''жb''b' ←
  'eb''b''lb''b''ab''b''eb''b''mb''b''ьb''b''йb'' b''pb''b''об''b''дб''b''иб''b''tb'' ←
  b''eb''b''lb''b''ьb'' b''yb''b''зб''b''lb''b''ab''
* @return true b''eb''b''cb''b''lb''b''иб'' b''об''b''пb''b''eb''b''pb''b''ab''b''цb'' ←

```

```

        b''иб''b''яb'' b''вb''b''ыb''b''пb''b''об''b''лb''b''нb''b''еb''b''нb''b''аб''; ←
        false -
    * b''пb''b''об''b''лb''b''yb''b''чb''b''еb''b''нb''b''нb''b''ыb''b''йb'' b''yb''b'' ←
        'зb''b''еb''b''лb'' b''нb''b''еb'' b''мb''b''об''b''жb''b''еb''b''тb'' b''бb''b'' ←
        'ыb''b''тb''b''ьb'' b''pb''b''об''b''дb''b''иб''b''тb''b''еb''b''лb''b''еb''b''мb'' ←
        b''дb''b''лb''b''яb''
    * b''дb''b''аб''b''нb''b''нb''b''об''b''гb''b''об'' b''об''b''бb''b''ьb''b''еb''b'' ←
        'кb''b''тb''b''аб''.
    */
bool SetParent( SGNODE* aParent );

/**
    * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иб''b''яb'' GetNodeTypeName
    * b''вb''b''об''b''зb''b''вb''b''pb''b''аб''b''шb''b''аб''b''еb''b''тb'' b''тb''b'' ←
        'иб''b''пb'' b''yb''b''зb''b''лb''b''аб'' b''вb'' b''вb''b''иб''b''дb''b''еb'' b'' ←
        'тb''b''еb''b''кb''b''сb''b''тb''b''аб'' b''иб''b''лb''b''иб'' NULL, b''еb''b''сb'' ←
        b''лb''b''иб'' b''yb''b''зb''b''еb''b''лb'',
    * b''кb''b''аб''b''кb''b''иб''b''мb''-b''тb''b''об'' b''об''b''бb''b''pb''b''аб''b'' ←
        'зb''b''об''b''мb'', b''иб''b''мb''b''еb''b''еb''b''тb'' b''нb''b''еb''b''вb''b'' ←
        'еb''b''pb''b''нb''b''ыb''b''йb'' b''тb''b''иб''b''пb''.
    */
const char * GetNodeTypeName( S3D::SGTYPES aNodeType ) const;

/**
    * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иб''b''яb'' AddRefNode
    * b''дb''b''об''b''бb''b''аб''b''вb''b''лb''b''яb''b''еb''b''тb'' b''сb''b''сb''b'' ←
        'ыb''b''лb''b''кb''b''yb'' b''нb''b''аб'' b''сb''b''yb''b''шb''b''еb''b''сb''b'' ←
        'тb''b''вb''b''yb''b''юb''b''шb''b''иб''b''йb'' b''yb''b''зb''b''еb''b''лb'', b'' ←
        'кb''b''об''b''тb''b''об''b''pb''b''ыb''b''йb'' b''нb''b''еb'' b''пb''b''pb''b'' ←
        'иб''b''нb''b''аб''b''дb''b''лb''b''еb''b''жb''b''иб''b''тb''
    * (b''нb''b''еb'' b''яb''b''вb''b''лb''b''яb''b''еb''b''тb''b''сb''b''яb'' b''дb''b'' ←
        'об''b''чb''b''еb''b''pb''b''нb''b''иб''b''мb'') b''эb''b''тb''b''об''b''мb''b'' ←
        'yb'' b''об''b''бb''b''ьb''b''еb''b''кb''b''тb''b''yb''.
    *
    * @return true b''пb''b''pb''b''иб'' b''yb''b''сb''b''пb''b''еb''b''шb''b''нb''b''об'' ←
        b''мb'' b''зb''b''аб''b''вb''b''еb''b''pb''b''шb''b''еb''b''нb''b''иб''b''иб''
    */
bool AddRefNode( SGNODE* aNode );
bool AddRefNode( IFSG_NODE& aNode );

/**
    * b''Фb''b''yb''b''нb''b''кb''b''цb''b''иб''b''яb'' AddChildNode
    * b''дb''b''об''b''бb''b''аб''b''вb''b''лb''b''яb''b''еb''b''тb'' b''yb''b''зb''b'' ←
        'еb''b''лb'', b''яb''b''вb''b''лb''b''яb''b''юb''b''шb''b''иб''b''йb''b''сb''b'' ←
        'яb'' b''дb''b''об''b''чb''b''еb''b''pb''b''нb''b''нb''b''мb'' b''пb''b''об'' b'' ←
        'об''b''тb''b''нb''b''об''b''шb''b''еb''b''нb''b''иб''b''юb'' b''кb'' b''эb''b'' ←
        'тb''b''об''b''мb''b''yb'' b''об''b''бb''b''ьb''b''еb''b''кb''b''тb''b''yb''.
    *

```

```

    * @return true b''пб''b''пб''b''иб'' b''yb''b''cb''b''пб''b''eb''b''шб''b''нб''b''об'' ←
      b''мб'' b''зб''b''аб''b''вб''b''eb''b''рб''b''шб''b''eb''b''нб''b''иб''b''иб''
    */
    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''Кb''b''пб''b''аб''b''cb''b''cb'' IFSG_TRANSFORM
 * b''эб''b''тб''b''об'' b''об''b''бб''b''об''b''пб''b''об''b''чб''b''кб''b''аб'' b''дб''b'' ←
   'пб''b''яб'' b''cb''b''об''b''вб''b''мб''b''eb''b''cb''b''тб''b''иб''b''мб''b''об''b'' ←
   'cb''b''тб''b''иб'' b''cb'' b''бб''b''пб''b''об''b''кб''b''об''b''мб'' TRANSFORM b' ←
   'иб''b''зб'' b''гб''b''рб''b''аб''b''фб''b''аб'' b''cb''b''цб''b''eb''b''нб''b''ыб'' ←
   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''пб''b''пб''b''об''b''чб''b''иб''b''eb'' b''фб''b''yb''b''нб''b''кб''b''цб''b'' ←
       'иб''b''иб'' b''бб''b''аб''b''зб''b''об''b''вб''b''об''b''гб''b''об'' b''кб''b''пб'' ←
       b''аб''b''cb''b''cb''b''аб'', b''кб''b''об''b''тб''b''об''b''рб''b''ыб''b''eb'' b' ←
       'зб''b''дб''b''eb''b''cb''b''ьб'' b''нб''b''eb'' b''рб''b''аб''b''cb''b''cb''b''мб'' ←
       b''аб''b''тб''b''рб''b''иб''b''вб''b''аб''b''юб''b''тб''b''cb''b''яб'' */

```

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''Кb''b''пб''b''аб''b''cb''b''cb'' IFSG_SHAPE
 * b''об''b''бб''b''об''b''пб''b''об''b''чб''b''кб''b''аб'' b''дб''b''пб''b''яб'' b''кб''b'' ←
   'пб''b''аб''b''cb''b''cb''b''аб'' 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''иб''b''eb'' b''фb''b''yb''b''нb''b''кb''b''цb''b' ←
       'иб''b''иб'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''об'' 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''иб''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''иб''b''eb'' b''фb''b''yb''b''нb''b''кb''b''цb''b' ←
       'иб''b''иб'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''об'' 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''nb''b''nb''b''ыb''b''eb'' b''zb''b''дb''b''eb''b''cb''b''ьb'' */

/* b''cb''b''лb''b''eb''b''дb''b''yb''b''юb''b''щb''b''иб''b''eb'' b''фb''b''yb''b' ←
'nb''b''kb''b''цb''b''иб''b''иб'' b''nb''b''eb'' b''иб''b''cb''b''пb''b''об''b''лb'' ←
b''ьb''b''zb''b''yb''b''юb''b''тb''b''cb''b''яb'' b''yb''b''zb''b''лb''b''ab''b' ←
'mb''b''иб'' Appearance
b''иб'' b''mb''b''об''b''гb''b''yb''b''тb'' b''вb''b''об''b''zb''b''вb''b''pb''b' ←
'ab''b''щb''b''ab''b''тb''b''ьb'' b''kb''b''об''b''дb'' b''об''b''шb''b''иб''b' ←
'бb''b''kb''b''иб''

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''об'' b''об''b''бb''b''об''b''лb''b''об''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''лb''b''pb''b''об''b''чb''b''иб''b''eb'' b''fb''b''yb''b''нb''b''кb''b''цb''b' ←
       'иб''b''иб'' b''бb''b''ab''b''эb''b''об''b''эb''b''об''b''гb''b''об'' b''кb''b''лb'' ←
       b''ab''b''cb''b''cb''b''ab'', b''кb''b''об''b''тb''b''об''b''pb''b''ыb''b''eb'' b' ←
    */

```

```
'зб''b''дб''b''eb''b''cb''b''ьb'' b''нб''b''eb'' b''pb''b''ab''b''cb''b''cb''b''mb'' ←
b''ab''b''tb''b''pb''b''иб''b''вб''b''ab''b''юб''b''tb''b''cb''b''яб'' */
```

## ifsg\_coords.h

```
/**
 * b''Kb''b''лb''b''ab''b''cb''b''cb'' IFSG_COORDS
 * b''эб''b''tb''b''об'' b''об''b''бб''b''об''b''лb''b''об''b''чb''b''кб''b''ab'' b''дб''b'' ←
   'лb''b''яб'' 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''лb''b''pb''b''об''b''чb''b''иб''b''eb'' b''фb''b''yb''b''нб''b''кб''b''цb''b'' ←
       'иб''b''иб'' b''бб''b''ab''b''зб''b''об''b''вб''b''об''b''гб''b''об'' b''кб''b''лb'' ←
       b''ab''b''cb''b''cb''b''ab'' b''нб''b''eb'' b''лb''b''об''b''кб''b''ab''b''зб''b'' ←
       'ab''b''нб''b''нб''b''ьb''b''eb'' b''зб''b''дб''b''eb''b''cb''b''ьb'' */

    /* b''cb''b''лb''b''eb''b''дб''b''yb''b''юб''b''щб''b''иб''b''eb'' b''фb''b''yb''b'' ←
       'нб''b''кб''b''цb''b''иб''b''иб'' b''нб''b''eb'' b''иб''b''mb''b''eb''b''юб''b''tb'' ←
       b''зб''b''нб''b''ab''b''чb''b''eb''b''нб''b''иб''b''яб'' b''дб''b''лb''b''яб'' b'' ←
       'yb''b''зб''b''лb''b''об''b''вб''
       b''кб''b''об''b''об''b''pb''b''дб''b''иб''b''нб''b''ab''b''tb'' b''иб'' b''вб''b'' ←
       'cb''b''eb''b''гб''b''дб''b''ab'' b''вб''b''об''b''зб''b''вб''b''pb''b''ab''b'' ←
       'шб''b''ab''b''юб''b''tb'' b''зб''b''нб''b''ab''b''чb''b''eb''b''нб''b''иб''b'' ←
       'eb'' b''об''b''шб''b''иб''b''бб''b''кб''b''иб''

       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''об'' b''об''b''бb''b''об''b''лb''b''об''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''лb''b''pb''b''об''b''чb''b''иб''b''eb'' b''fb''b''yb''b''нb''b''кb''b''цb''b' ←
       'иб''b''иб'' b''бb''b''ab''b''зb''b''об''b''вb''b''об''b''гb''b''об'' b''кb''b''лb'' ←
       b''ab''b''cb''b''cb''b''ab'' b''нb''b''eb'' b''лb''b''об''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''yb''b''юb''b''шb''b''иб''b''eb'' b''fb''b''yb''b' ←
       'нb''b''кb''b''цb''b''иб''b''иб'' b''нb''b''eb'' b''иб''b''мb''b''eb''b''юb''b''тb'' ←
       b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иб''b''яb'' b''дb''b''лb''b''яb'' b' ←
       'yb''b''зb''b''лb''b''ab''
       b''иб''b''нb''b''дb''b''eb''b''кb''b''cb''b''об''b''вb'' b''кb''b''об''b''об''b' ←
       'pb''b''дb''b''иб''b''нb''b''ab''b''тb'' b''иб'' b''вb''b''cb''b''eb''b''гb''b' ←
       'дb''b''ab'' b''вb''b''об''b''зb''b''вb''b''pb''b''ab''b''шb''b''ab''b''юb''b' ←
       'тb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''иб''b''eb'' b''об''b''шb''b' ←
       'иб''b''бb''b''кb''b''иб''

    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''об'' b''об''b''бb''b''об''b''лb''b''об''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''nb''b''pb''b''ob''b''чb''b''иб''b''eb'' b''fb''b''yb''b''nb''b''kb''b''цb''b' ←
       'иб''b''иб'' b''бb''b''ab''b''зb''b''ob''b''вb''b''ob''b''гb''b''ob'' b''kb''b''лb'' ←
       b''ab''b''cb''b''cb''b''ab'' b''nb''b''eb'' b''пb''b''ob''b''kb''b''ab''b''зb''b' ←
       'ab''b''nb''b''nb''b''yb''b''eb'' b''зb''b''дb''b''eb''b''cb''b''ьb'' */

    /* b''cb''b''лb''b''eb''b''дb''b''yb''b''юb''b''щb''b''иб''b''eb'' b''fb''b''yb''b' ←
       'nb''b''kb''b''цb''b''иб''b''иб'' b''nb''b''eb'' b''иб''b''mb''b''eb''b''юb''b''тb'' ←
       b''зb''b''nb''b''ab''b''чb''b''eb''b''nb''b''иб''b''яb'' b''дb''b''лb''b''яb'' b' ←
       'yb''b''зb''b''лb''b''ab''
       b''вb''b''eb''b''kb''b''тb''b''ob''b''pb''b''ob''b''вb'' b''иб'' 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''шb''b' ←
       'ab''b''юb''b''тb'' b''зb''b''nb''b''ab''b''чb''b''eb''b''nb''b''иб''b''eb'' b' ←
       'ob''b''шb''b''иб''b''бb''b''kb''b''иб''

    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''kb''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 );

/* б''пб''б''пб''б''об''б''чб''б''иб''б''еб'' б''фб''б''yb''б''нб''б''кб''б''цб''б' ←
'иб''б''иб'' б''бб''б''аб''б''зб''б''об''б''вб''б''об''б''гб''б''об'' б''кб''б''пб'' ←
б''аб''б''сб''б''сб''б''аб'' б''нб''б''еб'' б''пб''б''об''б''кб''б''аб''б''зб''б' ←
'аб''б''нб''б''нб''б''yb''б''еб'' б''зб''б''дб''б''еб''б''сб''б''ьб'' */

/* б''сб''б''пб''б''еб''б''дб''б''yb''б''юб''б''шб''б''иб''б''еб'' б''фб''б''yb''б' ←
'нб''б''кб''б''цб''б''иб''б''иб'' б''нб''б''еб'' б''иб''б''мб''б''еб''б''юб''б''тб'' ←
б''зб''б''нб''б''аб''б''чб''б''еб''б''нб''б''иб''б''яб'' б''дб''б''пб''б''яб'' б' ←
'yb''б''зб''б''пб''б''аб''
б''вб''б''еб''б''кб''б''тб''б''об''б''рб''б''об''б''вб'' б''иб'' б''вб''б''сб''б' ←
'еб''б''гб''б''дб''б''аб'' б''вб''б''об''б''зб''б''вб''б''рб''б''аб''б''шб''б' ←
'аб''б''юб''б''тб'' б''зб''б''нб''б''аб''б''чб''б''еб''б''нб''б''иб''б''еб'' б' ←
'об''б''шб''б''иб''б''бб''б''кб''б''иб''

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
{
/**
 * б''фб''б''yb''б''нб''б''кб''б''цб''б''иб''б''яб'' GetLibVersion б''вб''б''об''б' ←
'зб''б''вб''б''рб''б''аб''б''шб''б''аб''б''еб''б''тб'' б''иб''б''нб''б''фб''б''об'' ←
б''рб''б''мб''б''аб''б''цб''б''иб''б''юб'' б''об'' б''вб''б''еб''б''рб''б''сб''б' ←
'иб''б''иб''
 * б''бб''б''иб''б''бб''б''пб''б''иб''б''об''б''тб''б''еб''б''кб''б''иб'' kicad_3dsg
 */
SGLIB_API void GetLibVersion( unsigned char* Major, unsigned char* Minor,
                             unsigned char* Patch, unsigned char* Revision );

// б''фб''б''yb''б''нб''б''кб''б''цб''б''иб''б''иб'' б''дб''б''пб''б''яб'' б''иб''б' ←
'зб''б''вб''б''пб''б''еб''б''чб''б''еб''б''нб''б''иб''б''яб'' б''иб''б''нб''б''фб''б' ←
''об''б''рб''б''мб''б''аб''б''цб''б''иб''б''иб'' б''пб''б''об'' б''yb''б''кб''б' ←
'аб''б''зб''б''аб''б''тб''б''еб''б''пб''б''яб''б''мб'' 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''fb''b''yb''b''nb''b''kb''b''cb''b''ib''b''яb'' CalcTriNorm
 * b''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''шb''b''ab''b''eb''b''tb'' b''nb''b' ←
  'ob''b''pb''b''mb''b''ab''b''lb''b''ьb''b''nb''b''yb''b''йb'' b''vb''b''eb''b''kb'' ←
  b''tb''b''ob''b''pb'' b''дb''b''lb''b''яb'' b''tb''b''pb''b''eb''b''yb''b''gb''b' ←
  'ob''b''lb''b''ьb''b''nb''b''ib''b''kb''b''ab'', b''ob''b''пb''b''иб''b''cb''b' ←
  'ab''b''nb''b''nb''b''ob''b''gb''b''ob'' b''vb''b''eb''b''pb''b''шb''b''иб''b''nb'' ←
  b''ab''b''mb''b''иб'' p1, p2, p3
 */
SGLIB_API SGVECTOR CalcTriNorm( const SGPOINT& p1, const SGPOINT& p2, const SGPOINT& p3 ←
    );

/**
 * b''fb''b''yb''b''nb''b''kb''b''cb''b''ib''b''яb'' WriteCache
 * b''zb''b''ab''b''пb''b''иб''b''cb''b''ьb''b''vb''b''ab''b''eb''b''tb'' b''дb''b' ←
  'eb''b''pb''b''eb''b''vb''b''ob'' SGNODE b''vb'' b''бb''b''иб''b''nb''b''ab''b' ←
  'pb''b''nb''b''yb''b''йb'' b''fb''b''ab''b''йb''b''lb'' b''kb''b''эb''b''шb''b' ←
  'ab''
 *
 * @param aFileName - b''nb''b''ab''b''zb''b''vb''b''ab''b''nb''b''иб''b''eb'' b''fb''b' ←
  ''ab''b''йb''b''lb''b''ab'' b''дb''b''lb''b''яb'' b''zb''b''ab''b''пb''b''иб''b' ←
  'cb''b''иб''
 * @param overwrite - b''дb''b''ob''b''lb''b''jb''b''eb''b''nb'' b''cb''b''ob''b''дb''b' ←
  ''eb''b''pb''b''jb''b''ab''b''tb''b''ьb'' b''иб''b''cb''b''tb''b''иб''b''nb''b' ←
  'yb'', b''eb''b''cb''b''lb''b''иб'' b''nb''b''yb''b''jb''b''nb''b''ob'' b''пb''b' ←
  'eb''b''pb''b''eb''b''zb''b''ab''b''пb''b''иб''b''cb''b''ab''b''tb''b''ьb'' b''cb'' ←
  b''yb''b''шb''b''eb''b''cb''b''tb''b''vb''b''yb''b''юb''b''шb''b''иб''b''йb'' b' ←
  'fb''b''ab''b''йb''b''lb''
 * @param aNode - b''lb''b''юb''b''бb''b''ob''b''йb'' b''yb''b''zb''b''eb''b''lb'' b' ←
  'иб''b''zb'' b''дb''b''eb''b''pb''b''eb''b''vb''b''ab'', b''kb''b''ob''b''tb''b' ←
  'ob''b''pb''b''yb''b''йb'' b''nb''b''yb''b''jb''b''nb''b''ob'' b''zb''b''ab''b' ←
  'пb''b''иб''b''cb''b''ab''b''tb''b''ьb''
 * @return true b''пb''b''pb''b''иб'' b''yb''b''cb''b''пb''b''eb''b''шb''b''nb''b''ob'' ←
  b''mb'' b''zb''b''ab''b''vb''b''eb''b''pb''b''шb''b''eb''b''nb''b''иб''b''иб''
 */
SGLIB_API bool WriteCache( const char* aFileName, bool overwrite, SGNODE* aNode,
    const char* aPluginInfo );

/**
 * b''fb''b''yb''b''nb''b''kb''b''cb''b''ib''b''яb'' ReadCache
 * b''cb''b''чb''b''иб''b''tb''b''ьb''b''vb''b''ab''b''eb''b''tb'' b''бb''b''иб''b' ←
  'nb''b''ab''b''pb''b''nb''b''yb''b''йb'' b''fb''b''ab''b''йb''b''lb'' b''kb''b' ←
  'эb''b''шb''b''ab'' b''иб'' b''cb''b''ob''b''zb''b''дb''b''ab''b''eb''b''tb'' b' ←

```

```

    'дб''б''еб''б''рб''б''еб''б''вб''б''об'' SGNODE
*
* @param aFileName - б''иб''б''мб''б''яб'' б''бб''б''иб''б''нб''б''аб''б''рб''б''нб''б ←
    'об''б''рб''б''об'' б''фб''б''аб''б''йб''б''лб''б''аб'' б''кб''б''эб''б''шб''б'' ←
    'аб'' б''дб''б''лб''б''яб'' б''сб''б''чб''б''иб''б''тб''б''ыб''б''вб''б''аб''б'' ←
    'нб''б''иб''б''яб''
* @return NULL б''пб''б''рб''б''иб'' б''сб''б''бб''б''об''б''еб'', б''вб'' б''сб''б'' ←
    'лб''б''yb''б''чб''б''аб''б''еб'' б''yb''б''сб''б''пб''б''еб''б''хб''б''аб'' - б' ←
    'вб''б''об''б''зб''б''вб''б''рб''б''аб''б''шб''б''аб''б''еб''б''тб'' б''yb''б''кб'' ←
    б''аб''б''зб''б''аб''б''тб''б''еб''б''лб''б''ьб'' б''нб''б''аб''
* б''yb''б''зб''б''еб''б''лб'' б''вб''б''еб''б''рб''б''хб''б''нб''б''еб''б''рб''б'' ←
    'об'' б''yb''б''рб''б''об''б''вб''б''нб''б''яб'' SCENEGRAPH;
* б''еб''б''сб''б''лб''б''иб'' б''пб''б''об''б''нб''б''аб''б''дб''б''об''б''бб''б'' ←
    'иб''б''тб''б''сб''б''яб'', б''эб''б''тб''б''об''б''тб'' б''yb''б''зб''б''еб''б'' ←
    'лб'' б''мб''б''об''б''жб''б''нб''б''об'' б''сб''б''вб''б''яб''б''зб''б''аб''б'' ←
    'тб''б''ьб'' б''сб'' б''об''б''бб''б''об''б''лб''б''об''б''чб''б''кб''б''об''б'' ←
    'йб'' IFSG_TRANSFORM
* б''сб'' б''пб''б''об''б''мб''б''об''б''шб''б''ьб''б''юб'' б''фб''б''yb''б''нб''б'' ←
    'кб''б''цб''б''иб''б''иб'' IFSG_TRANSFORM::Attach().
*/
SGLIB_API SGNODE* ReadCache( const char* aFileName, void* aPluginMgr,
    bool (*aTagCheck)( const char*, void* ) );

/**
* б''фб''б''yb''б''нб''б''кб''б''цб''б''иб''б''яб'' WriteVRML
* б''зб''б''аб''б''пб''б''иб''б''сб''б''ыб''б''вб''б''аб''б''еб''б''тб'' б''пб''б'' ←
    'еб''б''рб''б''еб''б''дб''б''аб''б''нб''б''нб''б''ыб''б''йб'' б''yb''б''зб''б''еб'' ←
    б''лб'' б''иб'' б''еб''б''рб''б''об'' б''дб''б''об''б''чб''б''еб''б''рб''б''нб''б'' ←
    'иб''б''еб'' б''yb''б''зб''б''лб''б''ыб'' б''вб'' б''фб''б''аб''б''йб''б''лб'' ←
    VRML2
*
* @param filename - б''иб''б''мб''б''яб'' б''фб''б''аб''б''йб''б''лб''б''аб'' б''дб''б'' ←
    'лб''б''яб'' б''зб''б''аб''б''пб''б''иб''б''сб''б''иб''
* @param overwrite - б''дб''б''об''б''лб''б''жб''б''еб''б''нб'' б''бб''б''ыб''б''тб''б'' ←
    'ьб'' б''yb''б''сб''б''тб''б''аб''б''нб''б''об''б''вб''б''лб''б''еб''б''нб'' б' ←
    'вб'' б''иб''б''сб''б''тб''б''иб''б''нб''б''yb'', б''чб''б''тб''б''об''б''бб''б'' ←
    'ыб'' б''пб''б''еб''б''рб''б''еб''б''зб''б''аб''б''пб''б''иб''б''сб''б''аб''б''тб'' ←
    б''ьб''
* б''сб''б''yb''б''шб''б''еб''б''сб''б''тб''б''вб''б''yb''б''юб''б''шб''б''иб''б''йб'' ←
    б''фб''б''аб''б''йб''б''лб'' VRML
* @param aTopNode - б''yb''б''кб''б''аб''б''зб''б''аб''б''тб''б''еб''б''лб''б''ьб'' б' ←
    'нб''б''аб'' б''об''б''бб''б''ьб''б''еб''б''кб''б''тб'' SCENEGRAPH, б''пб''б''рб''б'' ←
    'еб''б''дб''б''сб''б''тб''б''аб''б''вб''б''лб''б''яб''б''юб''б''шб''б''иб''б''йб'' ←
    б''сб''б''цб''б''еб''б''нб''б''yb'' VRML
* @param reuse - б''дб''б''об''б''лб''б''жб''б''еб''б''нб'' б''бб''б''ыб''б''тб''б'' ←
    'ьб'' б''yb''б''сб''б''тб''б''аб''б''нб''б''об''б''вб''б''лб''б''еб''б''нб'' б' ←
    'вб'' б''иб''б''сб''б''тб''б''иб''б''нб''б''yb'', б''дб''б''лб''б''яб'' б''иб''б'' ←
    'сб''б''пб''б''об''б''лб''б''ьб''б''зб''б''об''б''вб''б''аб''б''нб''б''иб''б''яб''

```

```

* b''cb''b''vb''b''ob''b''yb''b''cb''b''tb''b''vb'' VRML DEF/USE
* @return true b''пб''b''рб''b''иб'' b''yb''b''cb''b''пб''b''eb''b''шб''b''нб''b''об'' ←
  b''мб'' b''зб''b''аб''b''vb''b''eb''b''рб''b''шб''b''eb''b''нб''b''иб''b''иб''
*/
SGLIB_API bool WriteVRML( const char* filename, bool overwrite, SGNODE* aTopNode,
                          bool reuse, bool renameNodes );

// b''пб''b''рб''b''иб''b''мб''b''eb''b''чб''b''Ab''b''нб''b''иб''b''Eb'' : b''cb''b'' ←
  'лб''b''eb''b''дб''b''yb''b''юб''b''щб''b''иб''b''eb'' b''фб''b''yb''b''нб''b''кб''b'' ←
  ''цб''b''иб''b''иб'' b''иб''b''cb''b''пб''b''об''b''лб''b''ьб''b''зб''b''yb''b''юб'' ←
  b''tb''b''cb''b''яб'' b''cb''b''об''b''vb''b''мб''b''eb''b''cb''b''tb''b''нб''b'' ←
  'об'' b''дб''b''лб''b''яб'' b''cb''b''об''b''зб''b''дб''b''аб''b''нб''b''иб''b''яб'' ←
  b''cb''b''бб''b''об''b''рб''b''кб''b''иб'' VRML,
// b''кб''b''об''b''tb''b''об''b''рб''b''аб''b''яб'' b''мб''b''об''b''жб''b''eb''b'' ←
  'tb'' b''иб''b''cb''b''пб''b''об''b''лб''b''ьб''b''зб''b''об''b''vb''b''аб''b''tb''b'' ←
  ''ьб'' b''нб''b''eb''b''cb''b''кб''b''об''b''лб''b''ьб''b''кб''b''об'' b''об''b'' ←
  'бб''b''ьб''b''eb''b''кб''b''tb''b''об''b''vb'' b''дб''b''лб''b''яб'' b''кб''b''аб'' ←
  b''жб''b''дб''b''об''b''гб''b''об'' SG*-b''кб''b''лб''b''аб''b''cb''b''cb''b''аб''.
// b''Vb'' b''об''b''бб''b''ьб''b''чб''b''нб''b''об''b''мб'' b''cb''b''лб''b''yb''b'' ←
  'чб''b''аб''b''eb'' b''дб''b''об''b''лб''b''жб''b''нб''b''об'' b''бб''b''ьб''b''tb'' ←
  b''ьб'' b''tb''b''аб''b''кб'' :
// 1) b''vb''b''ьб''b''зб''b''об''b''vb'' b''фб''b''yb''b''нб''b''кб''b''цб''b''иб''b'' ←
  'иб'' 'ResetNodeIndex()' b''дб''b''лб''b''яб'' b''cb''b''бб''b''рб''b''об''b''cb''b'' ←
  'аб'' b''гб''b''лб''b''об''b''бб''b''аб''b''лб''b''ьб''b''нб''b''об''b''гб''b''об'' ←
  b''иб''b''нб''b''дб''b''eb''b''кб''b''cb''b''аб'' b''иб''b''мб''b''eb''b''нб'' b'' ←
  'yb''b''зб''b''лб''b''об''b''vb'';
// 2) b''дб''b''лб''b''яб'' b''кб''b''аб''b''жб''b''дб''b''об''b''гб''b''об'' b''yb''b'' ←
  'кб''b''аб''b''зб''b''аб''b''tb''b''eb''b''лб''b''яб'' b''мб''b''об''b''дб''b''eb''b'' ←
  ''лб''b''иб'', b''пб''b''об''b''лб''b''yb''b''чб''b''eb''b''нб''b''нб''b''об''b'' ←
  'гб''b''об'' b''cb'' b''пб''b''об''b''мб''b''об''b''щб''b''ьб''b''юб'' 'S3D_CACHE-> ←
  Load()',
// b''eb''b''дб''b''иб''b''нб''b''об''b''жб''b''дб''b''ьб'' b''vb''b''ьб''b''зб''b'' ←
  'ьб''b''vb''b''аб''b''eb''b''tb''b''cb''b''яб'' 'RenameNodes()'. b''Tb''b''аб''b'' ←
  'кб''b''иб''b''мб'' b''об''b''бб''b''рб''b''аб''b''зб''b''об''b''мб'' b''дб''b''об'' ←
  b''cb''b''tb''b''иб''b''гб''b''аб''b''юб''b''tb'' b''tb''b''об''b''гб''b''об'', b'' ←
  'чб''b''tb''b''об''b''бб''b''ьб''
// b''vb''b''cb''b''eb'' b''yb''b''зб''b''лб''b''ьб'', b''пб''b''об''b''лб''b''yb''b'' ←
  ''чб''b''eb''b''нб''b''нб''b''ьб''b''eb'' b''иб''b''зб'' b''vb''b''ьб''b''xb''b'' ←
  'об''b''дб''b''нб''b''об''b''гб''b''об'' b''фб''b''аб''b''йб''b''лб''b''аб'', b'' ←
  'иб''b''мб''b''eb''b''лб''b''иб'' b''yb''b''нб''b''иб''b''кб''b''аб''b''лб''b''ьб''b'' ←
  ''нб''b''ьб''b''eb'' b''иб''b''мб''b''eb''b''нб''b''аб''.
// b''Фб''b''yb''b''нб''b''кб''b''цб''b''иб''b''яб'' RenameNodes() b''пб''b''eb''b'' ←
  'рб''b''eb''b''иб''b''мб''b''eb''b''нб''b''об''b''vb''b''ьб''b''vb''b''аб''b''eb''b'' ←
  'tb'' b''пб''b''об''b''лб''b''yb''b''чб''b''eb''b''нб''b''нб''b''ьб''b''йб'' b''yb'' ←
  b''зб''b''eb''b''лб'' b''иб'' b''vb''b''cb''b''eb'' b''eb''b''гб''b''об'' b''дб''b'' ←
  'об''b''чб''b''eb''b''рб''b''нб''b''иб''b''eb''
// b''yb''b''зб''b''лб''b''ьб''. b''Cb''b''vb''b''яб''b''зб''b''аб''b''нб''b''нб''b'' ←
  'ьб''b''eb'' b''yb''b''зб''b''лб''b''ьб'' b''об''b''cb''b''tb''b''аб''b''юб''b''tb'' ←

```

```

b''cb''b''яb'' b''бb''b''eb''b''зb'' b''иб''b''зb''b''mb''b''eb''b''нb''b''eb''b' ←
'нb''b''иб''b''йb''. b''Иb''b''cb''b''пb''b''об''b''лb''b''ьb''b''зb''b''об''b''вb'' ←
b''аб''b''нb''b''иб''b''eb'' b''yb''b''kb''b''аб''b''зb''b''аб''b''тb''b''eb''b' ←
'лb''b''яb'',
// b''пb''b''об''b''лb''b''yb''b''чb''b''eb''b''нb''b''нb''b''об''b''гb''b''об'' b' ←
'иб''b''зb'' b''fb''b''yb''b''нb''b''kb''b''цb''b''иб''b''иб'' S3DCACHE->Load(), b' ←
'пb''b''об''b''зb''b''вb''b''об''b''лb''b''яb''b''eb''b''тb'' b''yb''b''бb''b''eb'' ←
b''дb''b''иб''b''тb''b''ьb''b''cb''b''яb'' b''вb'' b''тb''b''об''b''mb'', b''чb''b' ←
'тb''b''об''
// b''вb''b''cb''b''eb'' b''дb''b''об''b''чb''b''eb''b''pb''b''нb''b''иб''b''eb'' b' ←
'yb''b''зb''b''лb''b''ьb'', b''пb''b''об'' b''об''b''тb''b''нb''b''об''b''шb''b' ←
'eb''b''нb''b''иб''b''юb'' b''kb'' b''пb''b''об''b''cb''b''лb''b''eb''b''дb''b''нb'' ←
b''eb''b''mb''b''yb'', b''бb''b''yb''b''дb''b''yb''b''тb'' b''иб''b''mb''b''eb''b' ←
'тb''b''ьb'' b''yb''b''нb''b''иб''b''kb''b''аб''b''лb''b''ьb''b''нb''b''ьb''b''eb'' ←
b''иб''b''mb''b''eb''b''нb''b''аб'';
// 3) b''eb''b''cb''b''лb''b''иб'' SG*-b''дb''b''eb''b''pb''b''eb''b''вb''b''об'' b' ←
'cb''b''об''b''зb''b''дb''b''аб''b''нb''b''об'' b''нb''b''eb''b''зb''b''аб''b''вb''b' ←
'иб''b''cb''b''иб''b''mb''b''об'' b''об''b''тb'' S3DCACHE->Load(), b''тb''b''об'' b' ←
'пb''b''об''b''лb''b''ьb''b''зb''b''об''b''вb''b''аб''b''тb''b''eb''b''лb''b''ьb''
// b''дb''b''об''b''лb''b''жb''b''eb''b''нb'' b''вb''b''ьb''b''зb''b''вb''b''аб''b' ←
'тb''b''ьb'' RenameNodes() b''kb''b''аб''b''kb'' b''пb''b''об''b''лb''b''об''b''жb'' ←
b''eb''b''нb''b''об'', b''чb''b''тb''b''об''b''бb''b''ьb'' b''об''b''бb''b''eb''b' ←
'cb''b''пb''b''eb''b''чb''b''иб''b''тb''b''ьb'' b''вb''b''cb''b''eb'' b''yb''b''зb'' ←
b''лb''b''ьb''
// b''yb''b''нb''b''иб''b''kb''b''аб''b''лb''b''ьb''b''нb''b''ьb''b''mb''b''иб'' b' ←
'иб''b''mb''b''eb''b''нb''b''аб''b''mb''b''иб'';
// 4) b''cb''b''об''b''зb''b''дb''b''аб''b''тb''b''ьb'' b''cb''b''тb''b''pb''b''yb''b' ←
'kb''b''тb''b''yb''b''pb''b''yb'' b''cb''b''бb''b''об''b''pb''b''kb''b''иб'' b''пb'' ←
b''yb''b''тb''b''ёb''b''mb'' b''cb''b''об''b''зb''b''дb''b''аб''b''нb''b''иб''b' ←
'яb'' b''нb''b''об''b''вb''b''об''b''гb''b''об'' b''yb''b''зb''b''лb''b''аб'' ←
IFSG_TRANSFORM, b''kb''b''аб''b''kb''
// b''пb''b''об''b''лb''b''аб''b''гb''b''аб''b''eb''b''тb''b''cb''b''яb'' b''дb''b' ←
'лb''b''яb'' b''kb''b''аб''b''жb''b''дb''b''об''b''гb''b''об'' b''эb''b''kb''b''зb'' ←
b''eb''b''mb''b''пb''b''лb''b''яb''b''pb''b''аб'' b''kb''b''об''b''mb''b''пb''b' ←
'об''b''нb''b''eb''b''нb''b''тb''b''об''b''вb''; b''бb''b''аб''b''зb''b''об''b''вb'' ←
b''yb''b''юb'' b''mb''b''об''b''дb''b''eb''b''лb''b''ьb'' b''kb''b''об''b''mb''b' ←
'пb''b''об''b''нb''b''eb''b''нb''b''тb''b''аб'',
// b''вb''b''об''b''зb''b''вb''b''pb''b''аб''b''шb''b''аб''b''eb''b''mb''b''yb''b' ←
'юb'' b''fb''b''yb''b''нb''b''kb''b''цb''b''иб''b''eb''b''йb'' S3DCACHE->Load(), b' ←
'mb''b''об''b''жb''b''нb''b''об'' b''дb''b''об''b''бb''b''аб''b''вb''b''иб''b''тb''b' ←
'ьb'' b''kb'' b''дb''b''аб''b''нb''b''нb''b''об''b''mb''b''yb'' b''yb''b''зb''b' ←
'лb''b''yb''
// IFSG_TRANSFORM b''cb'' b''пb''b''об''b''mb''b''об''b''шb''b''ьb''b''юb'' ' ←
AddRefNode();
// 5) b''yb''b''бb''b''eb''b''дb''b''иб''b''тb''b''ьb''b''cb''b''яb'', b''чb''b''тb''b' ←
'об'' b''вb''b''cb''b''eb'' b''нb''b''об''b''вb''b''ьb''b''eb'' b''yb''b''зb''b' ←
'лb''b''ьb'' IFSG_TRANSFORM b''дb''b''об''b''бb''b''аб''b''вb''b''лb''b''eb''b''нb'' ←
b''ьb'' b''вb'' b''kb''b''аб''b''чb''b''eb''b''cb''b''тb''b''вb''b''eb'' b''дb''b' ←

```

```

'об''б''чб''б''еб''б''рб''б''нб''б''иб''б''хб''
//  б''кб'' б''уб''б''зб''б''лб''б''уб'' б''вб''б''еб''б''рб''б''хб''б''нб''б''еб''б' ←
'гб''б''об'' б''уб''б''рб''б''об''б''вб''б''нб''б''яб'' IFSG_TRANSFORM, б''пб''б' ←
'об''б''дб''б''гб''б''об''б''тб''б''об''б''вб''б''иб''б''вб'' б''еб''б''гб''б''об'', ←
б''тб''б''аб''б''кб''б''иб''б''мб'' б''об''б''бб''б''рб''б''аб''б''зб''б''об''б' ←
'мб'', б''кб''
//  б''дб''б''аб''б''лб''б''ьб''б''нб''б''еб''б''йб''б''шб''б''еб''б''мб''б''уб'' б' ←
'пб''б''еб''б''рб''б''еб''б''иб''б''мб''б''еб''б''нб''б''об''б''вб''б''аб''б''нб''б' ←
'иб''б''юб'' б''иб'' б''зб''б''аб''б''пб''б''иб''б''сб''б''иб'';
// 6) б''вб''б''ыб''б''зб''б''вб''б''аб''б''тб''б''ьб'' RenameNodes() б''дб''б''лб''б' ←
'яб'' б''уб''б''зб''б''лб''б''аб'' б''сб''б''бб''б''об''б''рб''б''кб''б''иб'' б' ←
'вб''б''еб''б''рб''б''хб''б''нб''б''еб''б''гб''б''об'' б''уб''б''рб''б''об''б''вб''б' ←
'нб''б''яб'';
// 7) б''вб''б''ыб''б''зб''б''вб''б''аб''б''тб''б''ьб'' WriteVRML() б''вб'' б''об''б' ←
'бб''б''ыб''б''чб''б''нб''б''об''б''мб'' б''пб''б''об''б''рб''б''яб''б''дб''б''кб''б' ←
''еб'', б''сб'' б''пб''б''аб''б''рб''б''аб''б''мб''б''еб''б''тб''б''рб''б''об''б' ←
'мб'' renameNodes = false,
//  б''чб''б''тб''б''об''б''бб''б''ыб'' б''зб''б''аб''б''пб''б''иб''б''сб''б''аб''б' ←
'тб''б''ьб'' б''вб''б''сб''б''юб'' б''сб''б''тб''б''рб''б''уб''б''кб''б''тб''б''уб'' ←
б''рб''б''уб'' б''сб''б''бб''б''об''б''рб''б''кб''б''иб'' б''вб'' б''об''б''дб''б' ←
'иб''б''нб'' VRML-б''фб''б''аб''б''йб''б''лб''.
// 8) б''вб''б''ыб''б''сб''б''вб''б''об''б''бб''б''об''б''дб''б''иб''б''тб''б''ьб'' б' ←
'пб''б''аб''б''мб''б''яб''б''тб''б''ьб'', б''уб''б''дб''б''аб''б''лб''б''иб''б''вб'' ←
б''вб''б''сб''б''еб'' IFSG_TRANSFORM б''пб''б''еб''б''рб''б''еб''б''мб''б''еб''б' ←
'нб''б''нб''б''ыб''б''еб'' б''иб'' б''об''б''бб''б''ьб''б''еб''б''кб''б''тб''б''ыб'' ←
б''пб''б''рб''б''об''б''чб''б''иб''б''хб''
//  SG*-б''кб''б''лб''б''аб''б''сб''б''сб''б''об''б''вб'', б''кб''б''об''б''тб''б' ←
'об''б''рб''б''ыб''б''еб'' б''бб''б''ыб''б''лб''б''иб'' б''сб''б''об''б''зб''б''дб'' ←
б''аб''б''нб''б''ыб'' б''иб''б''сб''б''кб''б''лб''б''юб''б''чб''б''иб''б''тб''б' ←
'еб''б''лб''б''ьб''б''нб''б''об'' б''дб''б''лб''б''яб'' б''зб''б''аб''б''пб''б''иб'' ←
б''сб''б''иб'' б''дб''б''аб''б''нб''б''нб''б''ыб''б''хб''.

/**
 * б''фб''б''уб''б''нб''б''кб''б''цб''б''иб''б''яб'' ResetNodeIndex
 * б''сб''б''бб''б''рб''б''аб''б''сб''б''ыб''б''вб''б''аб''б''еб''б''тб'' б''гб''б' ←
'лб''б''об''б''бб''б''аб''б''лб''б''ьб''б''нб''б''ыб''б''еб'' б''иб''б''нб''б''дб'' ←
б''еб''б''кб''б''сб''б''ыб'' SG*-б''кб''б''лб''б''аб''б''сб''б''сб''б''аб''
 *
 * @param aNode - б''мб''б''об''б''жб''б''еб''б''тб'' б''бб''б''ыб''б''тб''б''ьб'' б' ←
'лб''б''юб''б''бб''б''ыб''б''мб'' б''пб''б''об''б''дб''б''хб''б''об''б''дб''б''яб'' ←
б''шб''б''иб''б''мб'' SGNODE
 */
SGLIB_API void ResetNodeIndex( SGNODE* aNode );

/**
 * б''фб''б''уб''б''нб''б''кб''б''цб''б''иб''б''яб'' RenameNodes
 * б''пб''б''еб''б''рб''б''еб''б''иб''б''мб''б''еб''б''нб''б''об''б''вб''б''ыб''б''вб'' ←
б''аб''б''еб''б''тб'' б''уб''б''зб''б''еб''б''лб'' б''иб'' б''еб''б''гб''б''об'' б' ←

```



```

    'дб''б''об''б''чб''б''еб''б''рб''б''нб''б''иб''б''еб'' б''уб''б''зб''б''лб''б''ыб'' ←
    б''вб'' б''сб''б''об''б''об''б''тб''б''вб''б''еб''б''тб''б''сб''б''тб''б''вб''б'' ←
    'иб''б''иб'' б''сб'' б''тб''б''еб''б''кб''б''уб''б''шб''б''иб''б''мб''б''иб''
* б''зб''б''нб''б''аб''б''чб''б''еб''б''нб''б''иб''б''яб''б''мб''б''иб'' б''гб''б'' ←
    'лб''б''об''б''бб''б''аб''б''лб''б''ьб''б''нб''б''ыб''б''хб'' б''иб''б''нб''б''дб'' ←
    б''еб''б''кб''б''сб''б''об''б''вб'' SG*-б''кб''б''лб''б''аб''б''сб''б''сб''б''аб''
*
* @param aNode - б''уб''б''зб''б''еб''б''лб'' б''вб''б''еб''б''рб''б''хб''б''нб''б'' ←
    'еб''б''гб''б''об'' б''уб''б''рб''б''об''б''вб''б''нб''б''яб''
*/
SGLIB_API void RenameNodes( SGNODE* aNode );

/**
* б''Фб''б''уб''б''нб''б''кб''б''цб''б''иб''б''яб'' DestroyNode
* б''уб''б''дб''б''аб''б''лб''б''яб''б''еб''б''тб'' б''пб''б''еб''б''рб''б''еб''б'' ←
    'дб''б''аб''б''нб''б''нб''б''ыб''б''йб'' б''уб''б''зб''б''еб''б''лб'' SG*-б''кб''б'' ←
    'лб''б''аб''б''сб''б''сб''б''аб''. б''Эб''б''тб''б''аб'' б''Фб''б''уб''б''нб''б'' ←
    'кб''б''цб''б''иб''б''яб'' б''пб''б''об''б''зб''б''вб''б''об''б''лб''б''яб''б''еб'' ←
    б''тб'' б''бб''б''еб''б''зб''б''об''б''пб''б''аб''б''сб''б''нб''б''об''
* б''уб''б''дб''б''аб''б''лб''б''яб''б''тб''б''ьб'' SG*-б''уб''б''зб''б''лб''б''ыб'', ←
    б''нб''б''еб'' б''пб''б''рб''б''иб''б''бб''б''еб''б''гб''б''аб''б''яб'' б''кб'' б'' ←
    'сб''б''вб''б''яб''б''зб''б''ыб''б''вб''б''аб''б''нб''б''иб''б''юб'' б''сб'' б'' ←
    'сб''б''об''б''об''б''тб''б''вб''б''еб''б''тб''б''сб''б''тб''б''вб''б''уб''б''юб''б'' ←
    ''шб''б''еб''б''йб''
* IFSG*-б''об''б''бб''б''об''б''лб''б''об''б''чб''б''кб''б''об''б''йб''.
*/
SGLIB_API void DestroyNode( SGNODE* aNode );

// б''Пб''б''Рб''б''Иб''б''Мб''б''Еб''б''Чб''б''Аб''б''Нб''б''Иб''б''Еб'': б''сб''б'' ←
    'лб''б''еб''б''дб''б''уб''б''юб''б''шб''б''иб''б''еб'' б''Фб''б''уб''б''нб''б''кб''б'' ←
    ''цб''б''иб''б''иб'' б''об''б''бб''б''лб''б''еб''б''гб''б''чб''б''аб''б''юб''б''тб'' ←
    б''сб''б''об''б''зб''б''дб''б''аб''б''нб''б''иб''б''еб'' б''нб'' б''уб''б''дб''б'' ←
    'аб''б''лб''б''еб''б''нб''б''иб''б''еб'' б''сб''б''тб''б''рб''б''уб''б''кб''б''тб''б'' ←
    ''уб''б''рб''
// б''дб''б''аб''б''нб''б''нб''б''ыб''б''хб'' б''дб''б''лб''б''яб'' б''рб''б''еб''б'' ←
    'нб''б''дб''б''еб''б''рб''б''иб''б''нб''б''гб''б''аб''

/**
* б''Фб''б''уб''б''нб''б''кб''б''цб''б''иб''б''яб'' GetModel
* б''сб''б''об''б''зб''б''дб''б''аб''б''ёб''б''тб'' б''пб''б''рб''б''еб''б''дб''б'' ←
    'сб''б''тб''б''аб''б''вб''б''лб''б''еб''б''нб''б''иб''б''еб'' S3DMODEL б''дб''б'' ←
    'лб''б''яб'' aNode (б''чб''б''иб''б''сб''б''тб''б''ыб''б''еб'' б''дб''б''аб''б'' ←
    'нб''б''нб''б''ыб''б''еб'', б''бб''б''еб''б''зб'' б''пб''б''рб''б''еб''б''об''б'' ←
    'бб''б''аб''б''рб''б''зб''б''об''б''вб''б''аб''б''нб''б''иб''б''йб'')
*
* @param aNode - б''уб''б''зб''б''еб''б''лб'', б''кб''б''об''б''тб''б''об''б''рб''б'' ←
    'ыб''б''йб'' б''нб''б''уб''б''жб''б''нб''б''об'' б''пб''б''рб''б''еб''б''об''б'' ←
    'бб''б''рб''б''аб''б''зб''б''об''б''вб''б''аб''б''тб''б''ьб'' б''вб'' б''пб''б'' ←

```

```

        'pb''b''eb''b''db''b''cb''b''tb''b''ab''b''vb''b''lb''b''eb''b''nb''b''ib''b''eb'' ←
S3DMODEL
* @return - b''vb''b''ob''b''zb''b''vb''b''pb''b''ab''b''шb''b''ab''b''eb''b''tb'' b' ←
        'пb''b''pb''b''eb''b''db''b''cb''b''tb''b''ab''b''vb''b''lb''b''eb''b''nb''b''ib''b' ←
        ''eb'' S3DMODEL b''vb'' b''cb''b''lb''b''yb''b''чb''b''ab''b''eb'' b''yb''b''cb''b' ←
        'пb''b''eb''b''xb''b''ab'', b''иб''b''nb''b''ab''b''чb''b''eb'' - NULL
*/
SGLIB_API S3DMODEL* GetModel( SCENEGRAPH* aNode );

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' Destroy3DModel
* b''ob''b''cb''b''vb''b''ob''b''бb''b''ob''b''жb''b''db''b''ab''b''eb''b''tb'' b' ←
        'пb''b''ab''b''mb''b''яb''b''tb''b''ьb'', b''zb''b''ab''b''nb''b''иб''b''mb''b' ←
        'ab''b''eb''b''mb''b''yb''b''юb'' b''cb''b''tb''b''pb''b''yb''b''kb''b''tb''b''yb'' ←
        b''pb''b''ob''b''йb'' S3DMODEL b''иб'' b''cb''b''cb''b''ьb''b''lb''b''ab''b''eb''b' ←
        'tb'' b''yb''b''kb''b''ab''b''zb''b''ab''b''tb''b''eb''b''lb''b''ьb''
* b''cb''b''tb''b''pb''b''yb''b''kb''b''tb''b''yb''b''pb''b''ьb'' b''nb''b''ab'' NULL
*/
SGLIB_API void Destroy3DModel( S3DMODEL** aModel );

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' Free3DModel
* b''ob''b''cb''b''vb''b''ob''b''бb''b''ob''b''жb''b''db''b''ab''b''eb''b''tb'' b' ←
        'пb''b''ab''b''mb''b''яb''b''tb''b''ьb'', b''zb''b''ab''b''nb''b''иб''b''mb''b' ←
        'ab''b''eb''b''mb''b''yb''b''юb'' b''db''b''ab''b''nb''b''nb''b''ьb''b''mb''b''иб'' ←
        b''cb''b''tb''b''pb''b''yb''b''kb''b''tb''b''yb''b''pb''b''ьb'' S3DMODEL
*/
SGLIB_API void Free3DModel( S3DMODEL& aModel );

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' Free3DMesh
* b''ob''b''cb''b''vb''b''ob''b''бb''b''ob''b''жb''b''db''b''ab''b''eb''b''tb'' b' ←
        'пb''b''ab''b''mb''b''яb''b''tb''b''ьb'', b''zb''b''ab''b''nb''b''иб''b''mb''b' ←
        'ab''b''eb''b''mb''b''yb''b''юb'' b''db''b''ab''b''nb''b''nb''b''ьb''b''mb''b''иб'' ←
        b''cb''b''tb''b''pb''b''yb''b''kb''b''tb''b''yb''b''pb''b''ьb'' SMESH
*/
SGLIB_API void Free3DMesh( SMESH& aMesh );

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' New3DModel
* b''cb''b''ob''b''zb''b''db''b''ab''b''ёb''b''tb'' b''иб'' b''иб''b''nb''b''иб''b' ←
        'цb''b''иб''b''ab''b''lb''b''иб''b''zb''b''иб''b''pb''b''yb''b''eb''b''tb'' b''cb'' ←
        b''tb''b''pb''b''yb''b''kb''b''tb''b''yb''b''pb''b''yb'' S3DMODEL
*/
SGLIB_API S3DMODEL* New3DModel( void );

/**
* b''Фb''b''yb''b''nb''b''kb''b''цb''b''иб''b''яb'' Init3DMaterial

```

```

* b''иб''b''нб''b''иб''b''цб''b''иб''b''аб''b''лб''b''иб''b''зб''b''иб''b''рб''b''yb'' ←
  b''eb''b''тб'' b''cb''b''тб''b''рб''b''yb''b''кб''b''тб''b''yb''b''рб''b''yb'' ←
  SMATERIAL
*/
SGLIB_API void Init3DMaterial( SMATERIAL& aMat );

/**
* b''Фб''b''yb''b''нб''b''кб''b''цб''b''иб''b''яб'' Init3DMesh
* b''cb''b''об''b''зб''b''дб''b''аб''b''ёб''b''тб'' b''иб'' b''иб''b''нб''b''иб''b' ←
  'цб''b''иб''b''аб''b''лб''b''иб''b''зб''b''иб''b''рб''b''yb''b''eb''b''тб'' b''cb'' ←
  b''тб''b''рб''b''yb''b''кб''b''тб''b''yb''b''рб''b''yb'' 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.