**Общие правила библиотек**

Каждая функция в библиотеках имеет префикс с именем самой библиотеки которое формируется исходя из назначения. После следует символ \_ после которого (желательно) указывается имя объекта/подсистемы/подмодуля если имеется.

Все функции имя которых после \_ начинается с 0 можно вызывать без инициализации самой библиотеки, все остальные можно вызывать только после инициализации.

Набор стандартных функций с 0:

0GetVersion – возвращает номер версии текущей библиотеки.

0Create – инициализирует текущую библиотеку.

Функция Dbg\_Set переназначает стандартную функцию вывода сообщений, и доступна в любой момент времени, независимо от инициализации библиотеки.

Функции имя которых после \_ начинается с A являются начальными, и могут быть вызваны только после того как библиотека была инициализирована.

Набор стандартны функций с A:

Core\_AGetName – получить имя текущей инициализированной библиотеки, которое было указано при инициализации.

Core\_AKill – уничтожить инициализированную библиотеку.

Каждая инициализированная библиотека должна иметь имя, уникальное или нет задается в момент вызова инициализации исходя из нужд программиста.

Каждая библиотека также может называться подсистема, в некоторых случаях ядро.

Для подключения любой библиотеки (в виде заголовочного файла либо исходного кода) осуществляется посредством подключения в проект всего одного файла с кодом:

для подключения внешней библиотеки (dll) нужно подключить заголовочный файл с расширением h а также файл с для экспорта функций с расширением lib.

для подключения исходного кода библиотеки достаточно подключить только cpp файл с исходным кодом.

Имена заголовочного и файла с исходным кодом одинаковые и отражают назначение самой библиотеки. К примеру sxgcore – графическое ядро, sxinput – система ввода (клавиатура/мышь), sxpp – библиотека с постпроцессом и т.д.

**Начальный базовый интерфейс IBaseObject**

Начальным родителем всех интерфейсов является базовый интерфейс IBaseObject, который объявлен в файле gdefines.h и имеет следующий вид:

struct IBaseObject

{

virtual ~IBaseObject(){};

virtual void Release()=0;

};

**Система вывода сообщений в библиотеках**

Во всех библиотеках предусмотрена система вывода сообщений через единую функцию. По умолчанию все сообщения выводятся через стандартную функцию def\_report, объявленную в gdefines.h.

Стандартная функция имеет ограничение на длину сообщения которое регулируется макроопределением REPORT\_MSG\_MAX\_LEN объявленном в gdefines.h. При написании своего обработчика сообщений следует использовать данное ограничение.

Уровни критичности сообщений объявлены в gdefines.h:

#define REPORT\_MSG\_LEVEL\_NOTICE 0 /\* заметка \*/

#define REPORT\_MSG\_LEVEL\_WARRNING 1 /\* предупреждение \*/

#define REPORT\_MSG\_LEVEL\_ERROR -1 /\* ошибка, желательно вылетать \*/

В случае переназначения стандартной функции сообщений необходимо самостоятельно (в переназначаемой функции) обрабатывать уровни критичности, для примера можно использовать механизм стандартной функции.

Тип функции для сообщений:

typedef void (\*report\_func) (int level,const char\* format,...);

Каждая библиотека имеет функцию для переназначения стандартной функции сообщений на пользовательскую. Пример основного ядра:

SX\_LIB\_API void Core\_Dbg\_Set(report\_func rf);

Данная функция может вызываться в любой момент времени, инициализация библиотеки необязательна.

Для формирования строки вида полный\_путь\_до\_файла:номер\_текущей\_строки можно воспользоваться макроопределением gen\_msg\_location объявленной в gdefines.h.

**Ограничения на размеры**

#define CORE\_NAME\_MAX\_LEN 32 /\* максимальная длина имени объекта ядра/подсистемы \*/

#define OBJECT\_NAME\_MAX\_LEN 64 /\* максимальная длина имени абстрактного объекта \*/

#define CONFIG\_SECTION\_MAX\_LEN 64 /\* максимальная длина секции конфигурационного файла \*/

Под абстрактным объектом следует понимать обобщенные элементы движка, к примеру имя статической модели, имя источника света и прочее.