Инструментальные средства создания подключаемых модулей (plugins) к 3D Studio MAX

Пядушкин Д., Чикалов И.

Нижегородская Лаборатория Программных Технологий, ЗАО, Нижний Новгород, Россия

Аннотация

В работе представлен набор инструментальных средств для разработки подключаемых модулей к 3D Studio MAX. Библиотека классов расширяет 3D Studio MAX SDK и обеспечивает некоторую абстрактную модель подключаемого модуля, не зависящую от его типа. Использование библиотеки избавляет разработчика от выполнения большей части технической работы за счет автоматизации работы со ссылками, блоками параметров, создания карты сообщений. При работе с библиотекой возможно широкое использование библиотеки классов MFC. Библиотека дополнена специальным AppWizard'ом, генерирующим скелетный исходный код подключаемых модулей. Работа с инструментальными средствами проиллюстрирована на примере подключаемого модуля, выполняющего видеоэффекты.

Ключевые слова: *3D Studio MAX, подключаемый модуль, видеоэффекты, MFC, AppWizard.*

1. ВВЕДЕНИЕ.

Предлагается к рассмотрению набор инструментальных средств (AppWizard для MS Visual Studio и специализированная библиотека классов) предназначенный для разработки подключаемых модулей к пакету 3D Studio MAX.

Разработанная библиотека классов представляет абстрактную (не зависящую от конкретного типа подключаемого модуля) модель объекта-plugin'а, в которой максимально скрыты технические аспекты реализации. В состав библиотеки входят:

- набор стандартных для plugin'а экспортируемых на этапе загрузки функций;
- набор макросов, автоматизирующих написание классов-дескрипторов для экспортируемых plugin'ом объектов;
- классы-шаблоны, реализующие часть виртуальных функций технического назначения для классов экспортируемых объектов;
- набор макросов и классов-шаблонов, полностью инкапсулирующих блоки параметров (IParameterBlock) 3D Studio MAX и сводящих работу с ними к работе с CDialog-классами библиотеки MFC и ClassWizard'ом;
- макросы, позволяющие использовать технику message maps (карты сообщений) для обработки событий, специфичных для 3D Studio MAX.

Библиотека содержит также средства, значительно облегчающие или полностью автоматизирующие работу со ссылками на другие объекты (references), с системой undo/redo, процедуры клонирования, ввода-вывода и т.п.

Как важное достижение, следует отметить появившуюся возможность при создании подключаемых модулей широко использовать библиотеку классов MFC. Задачи построения пользовательского интерфейса, организации сложных структур данных, написание процедур ввода/вывода и т. п. теперь могут решаться средствами MFC. Особо следует отметить, что использование рассматриваемой библиотеки и библиотеки MFC позволит использовать ClassWizard, как средство автоматизации написания исходного кода подключаемого модуля.

Параллельно с разработкой библиотеки был создан AppWizard, генерирующий скелетный исходный код подключаемых модулей с использованием представляемой библиотеки классов, что отличает его от имеющихся аналогов.

2. ОПИСАНИЕ БИБЛИОТЕКИ.

В состав библиотеки входят классы, реализующие функциональность объектов-plugin'ов, набор специальных макросов и классов-шаблонов.

Создаваемый программистом класс-plugin должен, во-первых, наследовать одному из документированных в MAX SDK базовому классу, и, во-вторых, специальному классу XGLObjectFactory из рассматриваемой библиотеки. Класс XGLObjectFactory обеспечивает базовую функциональность т. н. XGL-объекта – работу с блоками параметров, ссылками, сообщениями и пр.

Для удобного использования данного класса библиотека предоставляет набор классов-шаблонов, определяющих необходимые виртуальные функции, специфичные для конкретного типа plugin'а.

Ниже приведен пример объявления класса-plugin'а с использованием шаблона.

```
class CMyObject :
 public TMAXPlugInObjBase<GeomObject> {
  DECLARE_OBJECT_ATTR("ObjName",
   "ObjCategory", ObjType/SuperClassID)
  // <Other stmnts & decls>
   . . .
};
```

В этом примере объявляется класс, реализующий plugin типа Procedural Object (процедурный объект) и

наследующий классу GeomObject из MAX SDK. Шаблон TMAXPlugInObjBase производит "настройку" класса XGLObjectFactory путем определения необходимых виртуальных функций как из класса XGLObjectFactory, так и из класса GeomObject. В состав библиотеки входят несколько подобных шаблонов, каждый из которых предназначен для определенного типа подключаемого модуля.

При объявлении класса-plugin'а обязательным является объявление атрибутов класса с помощью макроса DECLARE_OBJECT_ATTR. Атрибутами являются имя, категория и т.н. идентификатор суперкласса объекта.

Объявленные таким образом классы необходимо перечислить в *таблице экспорта* – таблице, содержащей все экспортируемые классы подключаемого модуля. На основе таблицы экспорта библиотека автоматически генерирует классы-дескрипторы и ряд вспомогательных функций, обязательных для подключаемого модуля 3D Studio MAX. Ниже приведен пример таблицы экспорта, содержащей две записи.

```
BEGIN_OBJECT_MAP(_VERSION, "Module descriptor")
    OBJECT_DECL(CMyObj1, OBJID(CLSID0))
    OBJECT_DECL(CMyObj1, OBJID(CLSID1))
END_OBJECT_MAP()
```

Предметом гордости разработчиков библиотеки является механизм работы с блоками параметров подключаемого модуля.

Согласно MAX SDK при создании блока параметров надо иметь несколько массивов структур, описывающих параметры создаваемого объекта – тип, количество параметров, некоторые флаги и пр. Плюс к этому с помощью еще одного довольно громоздкого массива описателей определяется интерфейс пользователя, необходимый для доступа к создаваемому блоку параметров.

Вместо этого предлагается более простой способ работы с блоками параметров. Стандартным способом программист создает класс-наследник от класса CDialog библиотеки MFC. Создает поля данных этого класса, соответствующие элементам блока параметров, и связывает их (поля данных) с элементами контроля диалогового окна через механизм DDX. В конструкторе класса-plugin'а необходимо перечислить элементы блока параметров в специальной *таблице ссылок.* В дальнейшем разработчик подключаемого модуля может работать с этим диалогом, как с обычным классом MFC – добавлять обработчики сообщений, редактировать ресурс и т. д. Таким образом, происходит полное сокрытие от разработчика механизма работы блоков параметров в 3D Studio MAX, вместо которого предлагается работать с полями данных CDialog-класса.

Добавим также, что данный подход распространяется, в том числе и на предопределенные в 3D Studio MAX «готовые» диалоговые панели. Приемы работы с такими диалогами полностью аналогичны тем, которые применяются к MFC-диалогам (создание, активация, удаление и пр.).

Работа с блоками параметров является частью общего механизма работы со ссылками (специфичными для 3D Studio MAX объектами). При использовании данной библиотеки разработчик чаще всего вообще не работает со ссылками непосредственно и избавлен от написания кода для хранения и управления ссылками. Более подробно работа со ссылками будет рассмотрена на конкретном примере модуля видеоэффектов.

При создании подключаемого модуля часто возникает необходимость получать и обрабатывать сообщения от других модулей и от самого MAX'а. Классы-plugin'ы из MAX SDK имеют специально предназначенную для этого виртуальную функцию NotifyRefChanged. Назначение данной функции – обрабатывать приходящие сообщения аналогично тому, как это делает функция окна Windows. Рассматриваемая библиотека классов позволяет строить карты сообщений, подобные используемым в MFC. Использование карт сообщений позволяет структурировать исходный код и сделать его более прозрачным и понятным.

При разработке библиотеки класса было уделено значительное внимание процедуре ввода-вывода данных подключаемого модуля в/из файла данных 3D Studio MAX. В библиотеке реализован механизм *упорядочивания* данных (serializing). Объект-plugin имеет виртуальный метод Serialize полностью аналогичный соответствующему методу класса CObject библиотеке MFC. Отображение архива на файл MAX полностью автоматизировано. Разработчик полностью изолирован от конкретного механизма ввода/вывода 3D Studio MAX.

Механизм упорядочивания данных широко используется в рассматриваемой технологии создания подключаемых модулей. В частности, механизм клонирования системы undo/redo также использует serializing.

Важным достижением при создании библиотеки является *унификация объектов*. Под унификацией здесь подразумевается то, что вне зависимости от типа plugin'а применяются одни и те же приемы и методы работы. К примеру, абсолютно разные подходы к организации интерфейса пользователя объекта-модификатора и объекта-текстуры в MAX SDK принимают одинаковую форму при использовании рассматриваемой библиотеки.

Вне зависимости от типа plugin'а все объекты имеют одинаковый программный интерфейс. При создании объекта вызывается метод `OnCreate()`, уведомление о входе или выходе в интерактивный режим редактирования параметров происходит вызовом методов `OnBeginPropertyChange()`/`OnEndPropertyChange()` соответственно, работа по созданию интерфейса пользователя сведена к работе с классами-потомками `CDialog` библиотеки MFC, предопределенные в 3D

Studio MAX диалоговые панели с точки зрения программиста идентичны MFC-диалогам.

3. РЕАЛИЗАЦИЯ ПОДКЛЮЧАЕМОГО МОДУЛЯ ВИДЕОЭФФЕКТОВ.

Технология использования представляемого инструментария рассмотрена на примере создания plugin'a видеоэффектов для плоских изображений, предметная часть которого была реализована с помощью библиотеки IPL (http://www.intel.com/). Подключаемый модуль создает результирующее изображение из нескольких исходных, используя один из нескольких реализованных видеоэффектов. На основе полученного изображения создается анимируемая процедурная 2D-текстура, которая может быть использована в качестве заднего плана сцены или как одна из карт при создании материала (например, карты отражения или цвета для диффузного освещения). Пользовательский интерфейс позволяет указать имена файлов, в которых хранятся исходные изображения, и настроить параметры видеоэффекта.

Процесс создания подключаемого модуля начинается с создания проекта с помощью специального AppWizard'a. На этом этапе автоматически создаются ресурсы и исходные тексты приложения, одинаковые для всех подключаемых модулей типа «процедурная текстура». Сгенерированный исходный код содержит объявление класса объекта-plugin'a, его (класса) реализацию, а так же объявление и реализацию классов диалоговых окон (наследников класса CDialog) для редактирования параметров plugin'a.

На следующем этапе в редакторе ресурсов создается диалоговая панель для управления параметрами видеоэффекта, и управляющие элементы связываются с параметрами посредством механизма DDX с помощью ClassWizard'a.

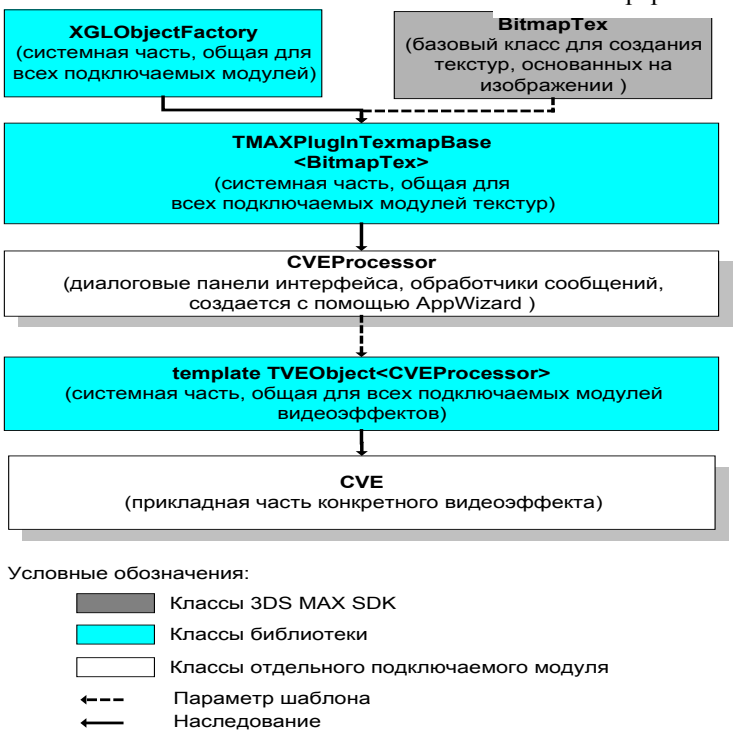
Затем от объявленного объекта наследуется класс через шаблон-модификатор, дополняющий объект «процедурная текстура» свойствами, общими для всех видеоэффектов. Для каждого параметра видеоэффекта в конструктор порожденного класса добавляется макрос, декларирующий тип параметра – статический или анимируемый. Переопределяется функция получения результирующего изображения из исходных. На этом процесс создания подключаемого модуля завершается.

На рис. 1 показана иерархия классов подключаемого модуля.

В шаблоне TVEObject реализованы функции работы с файлами исходных изображений, набор функций для рендеринга, обработчики некоторых событий системы и переопределены некоторые функции технического назначения. Применение шаблона класса взамен механизма наследования позволяет реализовать пользовательский интерфейс и обработку сообщений системы в «базовом» классе с помощью вышеупомянутых средств автоматизации. Представляется целесообразным использование шаблона при разработке серии подключаемых модулей для реализации общих свойств объектов.

Основное назначение подключаемого модуля – создание реалистичных сцен, обладающих невысокой вычислительной сложностью, за счет использования процедурных текстур взамен дорогостоящих средств (например, атмосферных эффектов), или сложной геометрии. Использование видеоэффектов в различных картах материала расширяет набор выразительных

Рис. 1. Иерархия классов подключаемого модуля.



средств 3D Studio MAX. Одна из областей применения подключаемого модуля – анимация различных трансформаций объектов сцены (возможно, с одновременным изменением геометрии).

Ниже на рисунках изображена сцена, на заднем плане которой используется видеоэффект «растворение» для смены времени суток.

Использование инструментальных средств позволило значительно сократить затраты ресурсов на разработку продукта. Для реализации конкретного видеоэффекта стало достаточно создать диалоговую панель параметров в стандартном редакторе ресурсов, определить их тип (статические или анимируемые), и переопределить функцию получения результирующего изображения из исходных. Таким образом, благодаря инструментальным средствам разработчик был практически полностью освобожден от технической работы по созданию подключаемого модуля.

Рис 1. t=0

Рис 2. t=50

Рис 3. t=100

Об авторах

Пядушкин Дмитрий, (e-mail: dim@nstl.nnov. ru),
Чикалов Игорь, студент 2-го курса магистратуры факультета Вычислительной математики и Кибернетики ННГУ.

Abstract

Instrumental Tools for 3D Studio MAX Plugins Development

D. Pyadushkin, I. Chikalov.

In the paper the set of instrumental tools for 3D Studio MAX plugins development is presented. The class library extends 3D Studio MAX SDK and provides some abstract model of pligin, which is independent of plugin type. Using of the library saves the developer from maior part of technical job via automatization of references and parameters block and creation of message map. Our class library allows to use MFC class library widely. The class library is complemented with special Application Wizard, which generates the skeleton of plugin's source code. The pligin performing several video effects is presented as the example of using of the instrumental tools.