2. Андерсон Т. Статистический анализ временных рядов. – М., 1976.
3. Бокс Дж., Дженкинс Г. Анализ временных рядов. – М., 1974.
4. Kharin Yu.S., Zenevich D.V. *Robastness of Autoregressive Forecasting under Misspecified Model.* Computer Data Analysis & Modeling. Proceedings of the V-th Int. Conf. Minsk, BSU, 1998, pp. 120–126.

КОНСОЛЬНОЕ УПРАВЛЕНИЕ РЕСУРСАМИ В ЗАДАЧАХ ПОСТРОЕНИЯ ТРЕХМЕРНОГО ИЗОБРАЖЕНИЯ

Бахтизин В.В., Круглов Д.Г.
Белорусский государственный университет информатики и радиоэлектроники
г.Минск, 220027, ул.П. Бровки 6

Аннотация: В данной статье рассматривается задача повышения качества библиотек визуализации трехмерной графики в реальном масштабе времени. Приводится метод ускорения и упрощения этапа тестирования и отладки графической модели, основанный на консольном управлении.

Ключевые слова: Библиотеки визуализации трехмерной графики, отладка и тестирование трехмерной модели, объектно-ориентированное построение библиотеки.

В последние годы системы визуализации трехмерной графики получили небывалое развитие. Бурный рост производительности персональных компьютеров сделал трехмерное моделирование доступным широкому кругу пользователей. В данный момент возможности трехмерной графики во всю используются в ряде инженерных, обучающих, информационных и демонстрационных систем. По этому особенно остро стоит вопрос создания надежных средств автоматизации разработки ПО (программного обеспечения) с применением трехмерной графики.

Перечислим последовательность создания такого ПО:

- проектирование трехмерной модели;
- программирование системы ввода-вывода;
- создание трехмерных графических ресурсов;
- тестирование и отладка модели;
- сдача в эксплуатацию.

В рассмотренной последовательности остановимся подробнее на этапе тестирования и отладки модели. Завершающая стадия работы над трехмерной моделью – одна из самых сложных. Связанно это с тем, что система ввода-

вывода и физическая модель уже запрограммированы. В такой ситуации очень тяжело находить и исправлять ошибки в ресурсах, или что еще хуже – в общей концепции построения модели. Необходимость возвращаться к этапу создания ресурсов для внесения самых мелких корректив, необходимость неоднократной компиляции (для больших по объему проектов это имеет особое значение) при обнаружении мелких ошибок отнимает массу времени. Тем более что с ресурсами чаще всего работает не программист, а дизайнер. В результате на отладку модели уходит большой объем времени.

Этот этап можно упростить, используя консольное управление. Анализ крупнейших мировых систем трехмерной визуализации показал эффективность применения такого подхода. Так, например, известные библиотеки LightTech, Geometry Box II, а также Quake Engine содержат в себе консоль.

Поясним подробнее, что означает термин «консольное управление» в рамках графической трехмерной системы. Из названия (консоль) становиться понятно, что это альтернативный интерфейс. Однако это интерфейс не с пользователем, а с разработчиком. Интерфейс на более низком уровне. Чаще всего он ограничен одной строкой ввода, и одной строкой вывода (возможна область вывода). Консольные команды предоставляют доступ ко всем элементам модели, к их свойствам и методам. Более того, они позволяют изменять параметры системы вывода, «на лету» меняя такие настройки, как разрешение, цветность, наличие освещения, текстурирование и пр. Таким образом, консоль является идеальным инструментом отладки модели. И выполняется отладка в процессе работы модели, а не отдельно.

Рассмотрим подробнее, в чем эффективность такого метода. Множество мелких ошибок модели появляются вследствие неточного или невнимательного размещения объектов в пределах сцены. Кроме того ошибки связывания объектов, ошибки параметров поведения объектов. Большинство из них достаточно очевидно исправляются, но сложность состоит в требуемом для этого времени. Изменения нужно производить во внешнем редакторе ресурсов, или, что сложнее, в коде программы с последующей компиляцией. Консоль позволяет исправить ошибку (изменить состояние модели) прямо в процессе работы. Проверить полученный результат и продолжить работу. По мере накопления информации об исправлениях, можно произвести общее исправление в ресурсах и коде. Причем, за счет предварительной проверки, можно быть уверенным в том, что эти исправления не повлекут нарушения работы системы.

В разрабатывасмой нами трехмерной графической системе визуализации, консоль реализована в большем объеме, чем в приведенных выше мировых аналогах. Фактически, при расширении возможностей консоль превращается в мини-транслятор, обрабатывающий команды языка управления ресурсами. В нашем случае, вся библиотека визуализации строится как иерархия классов (метод ООП). Каждый класс, за счет наследования, получает ряд функций по обработке запросов от консоли. Это могут быть вызовы методов класса, получение или установка значения переменных класса. В процессе работы все объекты системы получают уникальные имена и доступны по этим именам. Консоль, после поступления и интерпретации команды, находит нужный объект и передает ему интерпретированную команду. Весь процесс выглядит следующим образом:

- поступление команды;
- ее интерпретация консолью;
- определение необходимых параметров;
- нахождение объекта по его имени;
- передача объекту команды;
- передача объекту параметров;
- выполнение объектом команды.

В интерпретацию команды входит определение типа команды (вызов функции, создание/удаление объекта, получение значения переменной и т.д.). Далее, определение имени объекта и передаваемых ему параметров.

Нахождение любого объекта по имени является задачей отдельного системного класса. Отметим лишь, что это простейшая задача работы со списками. При передаче объекту команды, консоль «не знает» подробностей обработки команды. Каждый объект может выполнять свои специфические команды, и он сам проверит правильность всех параметров. Команды совпадают с публичными (public) функциями и переменными класса, таким образом, разработчику нет необходимости изучать новый для него язык консольных команд. Достаточно знать имена объектов модели (они доступны в ресурсах) и функции этих объектов. Фактически, такая консоль является одним из методов создания транслятора C/C++ в пределах конкретной задачи.

Подобный подход трансляции консольных команд реализован в трехмерном редакторе 3DStudioMax. Однако там он расширен до полного языка программирования, и на его создание было потрачен значительный объем временных и людских ресурсов. Приведенный выше вариант консоли отличается тем,

что его реализация не отнимает дополнительного времени у разработчика, так как исполнение команд в объектах – это простое перенаправление параметров на уже описанные в объекте методы. А программирование консоли требует незначительных усилий.

В дальнейшем, лишь незначительно изменив условия поступления команд, можно обрабатывать командные файлы. Таким образом, возможна автоматическая инициализация системы, что значительно проще программной, так как позволяет изменять параметры после компиляции программы. Кроме того, возможно создание тестовых последовательностей, отработка которых каждый раз при старте системы гарантирует ее работоспособность. Реализация этих дополнений значительно упростит работу разработчиков программных модулей многократного использования.

Реализовать консольное управление можно не только в графических системах визуализации. В большинстве систем, связанных с обработкой сложных моделей, хранящихся во внешних ресурсах может возникнуть необходимость приостановить работу системы, изменить параметры системы или модели и запустить ее опять. По мере накопления информации об ошибках и неточностях, а также точных методах их исправления производится перекомпиляция. Такой подход сокращает временные издержки на отладку любой системы.

ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Карасик Е. А.

Белорусский государственный университет информатики и радиоэлектроники

г. Минск, ул. П. Бровки, 6

Аннотация: обзорная статья по основным методам и стратегиям тестирования программного обеспечения.

Ключевые слова: тестирование; программное обеспечение;

Одними из основных методов повышения надежности ПО, являются тестирование, отладка и верификация, которые состоят в проверке КП на соответствие заданной спецификации [1].

Тестирование программного обеспечения охватывает целый ряд видов деятельности. Сюда входят постановка задачи для теста, проектирование, написание тестов, тестирование тестов и, наконец, выполнение тестов, и изучение результатов тестирования. Решающую роль играет проектирование тестов. Существует целый спектр подходов к проектированию тестов [2].