При этом используется 13 классов, из которых все кроме одного, соответствующего костям, порождают по одному объекту. Автоматы в проекте реализованы, как методы класса – объект («тело») и управляющий им автомат («душа») реализованы совместно.

Программа использует несколько потоков для обеспечения возможности одновременного движения костей. При этом в одном потоке работает «мозг» и синхронизатор, а для каждой из костей создается отдельный поток.

Применение автоматного подхода упрощает внесение изменений в проект, что позволит сравнительно «безболезненно» повысить его функциональность в следующих версиях программы. Проект размещен на сайте http://is.ifmo.ru в разделе «Проекты».

Работа выполнена при поддержке Российского фонда фундаментальных исследований по гранту № 02-07-90114 «Разработка технологии автоматного программирования».

Литература

1. Ваткин С. Скелетная анимация. http://www.gamedev.ru/coding/10917.shtml.

2. Шалыто А.А., Туккель Н.И. SWITCH-технология – автоматный подход к созданию программного обеспечения "реактивных" систем //Программирование. 2001. №5. http://is.ifmo.ru.

Ошибка! Указано неверное имя файла.

РЕАЛИЗАЦИЯ КОНЕЧНЫХ АВТОМАТОВ С ИСПОЛЬЗОВАНИЕМ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Г.А.Корнеев, А.А.Шалыто
Санкт-Петербургский государственный институт точной механики и оптики (технический университет)
Тел.: (812) 350-47-58, e-mail: kgeorgiy@rain.ifmo.ru

В работах [1, 2] была предложена концепция программирования с явным выделением состояний, базирующаяся на конечных автоматах. В настоящей работе предлагается метод реализации конечных автоматов с использованием объектно-ориентированного программирования.

В отличие от паттерна “State”, описанного в работе [3], предлагается реализовывать конечный автомат в виде одного класса (с использованием оператора “switch”), а не как набор классов, представляющих отдельные состояния автомата. При этом вся логика работы автомата концентрируется в одном месте, что упрощает отладку программы и внесение изменений при повторном использовании.

Для реализации унифицированного управления конечным автоматами и автоматизации генерации протоколов работы программ удобно вынести соответствующую им логику в базовый класс “BaseAutomata”, от которого будут наследоваться конкретные конечные автоматы.

Класс “BaseAutomata” должен предоставлять конечному автомату следующую функциональность:

- чтение и запись текущего состояния автомата;
- вызов функции переходов автомата;
- вызов функции, выполняющей действия, осуществляемые в состояниях (для автоматов Мура и смешанных автоматов).

При необходимости класс “BaseAutomata” может осуществлять запись в протокол работы программы при вызове функции переходов автомата и изменении его состояния (в функции изменения состояния автомата).

Для внешних модулей класс “BaseAutomata” должен представлять следующий функции:

- чтение номера текущего состояния автомата;
- чтение название текущего состояния (может применяться для генерирациии протоколов работы программы);
- генерация сообщений об изменении состояния автоматов (через “call-back” функции или очередь событий).
- вызов функции переходов автомата.

Реализация класса “BaseAutomata” и пример его использования приведены на сайте http://is.ifmo.ru в разделе “Тезисы”.

Предложенный подход позволяет совместить достоинства автоматного и объектно-ориентированного подходов. При этом функция переходов содержит только реализацию логики работы автомата.

Работа выполнена при поддержке Российского фонда фундаментальных исследований по гранту №02-07-90114 “Разработка технологии автоматного программирования”.

Литература

1. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.

2. Шалыто А.А., Туккель Н.И. Программирование с явным выделением состояний //Мир ПК. 2001. N8, N9.

3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001.

Ошибка! Указано неверное имя файла.

АВТОМАТНАЯ РЕАЛИЗАЦИЯ ИНТЕРАКТИВНЫХ СЦЕНАРИЕВ ОБРАЗОВАТЕЛЬНОЙ АНИМАЦИИ

М.А.Мазин, В.Г.Парфенов, А.А.Шалыто
Санкт-Петербургский государственный институт точной механики и оптики (технический университет)
Тел.: (812) 315-61-59, e-mail: mazine@rain.ifmo.ru

В настоящее время в образовательном процессе все чаще применяется анимация, которая позволяет наглядно представить изучаемый материал. Для предоставления широкого доступа к обучающим материалам целесообразно размещать их в сети Интернет. Наиболее пригодным средством для создания анимации в этом случае является пакет «Macromedia Flash».

Как правило, созданию анимации предшествует написание текста сценария, на основе которого впоследствии создается раскадровка. В образовательных приложениях целесообразно использовать интерактивную анимацию – анимацию, на развитие сюжета которой пользователь может влиять. В связи с этим в отличие от линейных сценариев мультфильмов возникает задача программирования нелинейных сценариев.

В настоящей работе предложен подход к разработке и реализации нелинейных сценариев с помощью интерактивной анимации на базе рассматриваемого пакета. Основу этого подхода составляет формализация сценариев и их формальная реализация на языке «ActionScript» [1]. Метод основан на применении автоматного подхода [2, 3].

Предлагаемый подход состоит из следующих шагов.

1. По вербальному заданию (тексту сценария) строится его математическая модель в виде графа переходов конечного автомата. Для этого сюжет разбивается на сцены – статические изображения в точках, в которых пользователь осуществляет ввод данных. Каждая сцена определяет состояние автомата, в котором указанное изображение формируется как выходное действие.

При этом пользовательский ввод генерирует входные переменные и порождает события, с которыми вызывается автомат. В зависимости от типа события и от значений входных переменных вызываемый автомат может осуществлять переходы.

Анимация, связанная с задачей, составляет совокупность дополнительных выходных действий, которые выполняются при переходах автомата.

Кроме того, при переходах выполняются выходные действия, связанные с управлением средой исполнения, например, таким действием является завершение работы флэш-плеера.

Изложенное позволяет построить схему связей автомата, которая описывает его интерфейс. При этом автомат управляет статическими изображениями, анимацией и средой исполнения.

2. По построенному графу переходов формально и изоморфно строится текст подпрограммы, которая реализует его логику на основе конструкции «switch» языка «ActionScript». Функции выходных действий и входных переменных на этом этапе заменяются функциями-заглушками. Обработчики событий выделяют из потока обрабатываемые события и вызывают с этими событиями автомат, который в зависимости от своего текущего состояния и значений входных переменных может осуществлять переходы и выходные действия.

3. После реализации графа переходов программист пишет программные модули, соответствующие выходным действиям, входным переменным и обработчикам событий. В то же время, руководствуясь схемой связей автомата, художник «рисует» статические изображения и выполняет раскадровку анимации.

Тем самым создается программа, состоящая из обработчиков событий, указанной выше конструкции «switch» и функций, которые вызываются из этой конструкции.

4. Далее программист связывает функции выходных действий с анимацией и статическими изображениями, созданными художником.

5. Отладка поведения программы в рамках предлагаемого подхода осуществляется с помощью протоколов (логов), для построения которых, вводятся функции протоколирования, вызываемые из конструкции «switch», функций выходных действий, входных переменных и обработчиков событий.

Работа выполнена при поддержке Российского фонда фундаментальных исследований по гранту 02-07-90114 «Разработка технологии автоматного программирования».

Литература

1. Сандерс Б. Flash ActionScript. СПб.: Питер. 2001.