УДК 004.9

А.А. Мавзютов, М.В. Черненький

МЕТОД РЕАЛИЗАЦИИ ОБУЧАЮЩИХ ВИРТУАЛЬНЫХ ТРЕХМЕРНЫХ ТРЕНАЖЕРОВ

При описании процессов со сложной нелинейной логикой исследователь обычно использует известные методологии описаний в виде диаграмм последовательностей и конечных автоматов. Чтобы описать процессы в системах реального времени, необходимо учесть возможность фиксации и реагирования системы на внешние и внутренние события. При этом переход от теоретического описания процесса в виде диаграмм конечного автомата к его программной реализации зачастую затруднен вследствие неполного соответствия средств проектирования и программирования. Используемый в работе метод событийного конечного автомата основан на подходе, реализуемом функциональными языками и в современных моделях программирования типа .Net Framework.

E-mail: chernen@bmstu.ru

***Ключевые слова*:** *конечный автомат, событийный автомат, разработка учебных тренажеров, 3D-моделирование, моделирование, искусственный интеллект.*

Модернизированный конечный автомат можно определить как циклически активизируемый набор альтернативных функций [1]:

$$A = (F, x, y, f_{cur}, f_1), \quad (1)$$

где x — входная переменная с допустимыми значениями из множества $I = \{1, 2, 3, \ldots, m, \ldots, M\}$; y — выходная переменная с допустимыми значениями из множества $O = \{1, 2, 3, \ldots, n, \ldots, N\}$; $F = \{f_1, \ldots, f_k, \ldots, f_K\}$ — множество альтернативных функций состояний, отображающих множество I в O и множество I в F. Другими словами, каждая из функций f_k содержит две подфункции:

1) выходов f_{Ok}, ставит в соответствие некоторому значению входа m значение выхода n;

2) переходов f_{Sk}, вычисляет функцию состояния для следующего цикла из множества F :

$$y = f_{Ok}(x), f_{Ok}(x) \in \mathrm{O}; f_{cur} = f_{Sk}(x), f_{Sk}(x) \in F,$$

$f_{cur} \in F$ — текущая функция состояния автомата для рассматриваемого момента времени; $f_1 \in F$ — начальная функция состояния автомата, текущая функция состояния при начале работы.

Актуальное полное описание конечного автомата задается его текущей функцией состояния f_{cur} для рассматриваемого момента времени (цикла).

Процесс и событийный полиморфизм. Определим функции состояния с учетом модели модернизированного конечного автомата (1) $F = \{f_1, \ldots, f_k, \ldots, f_K\}$, как множество альтернативных функций состо-

яния, отображающих входную переменную x в выходную переменную y ($y = f_{Ok}(x)$, $f_{Ok}(x) \in O$) и переменную x в модель F ($f_{cur} = f_{Sk}(x)$, $f_{Sk}(x) \in F$).

При данном представлении нет причин жестко привязывать функции состояния к уникальным переменным, как в модели конечного автомата. Переменные для каждой функции состояния могут быть уникальными. Таким образом, переменные могут привязываться исключительно к функции состояния.

С учетом изложенного выше можно ввести понятие «процесс». ***Процесс*** — полиморфная функция, совокупность альтернативных функций состояния, представляемых в программе как единая неделимая сущность. При вызове этого процесса выполняется только одна из составляющих его функций состояния — текущая функция состояние. Дополнительно к этому любой процесс снабжен индивидуальными часами — счетчиком времени, который работает постоянно и обнуляется только при смене текущей функции состояния.

Следовательно, процесс p_i задается четверкой элементов [2], которые отражают статическую и динамическую информацию о процессе:

$$p_i = (F_i,\ f_i^1,\ f_{i\,cur},\ T_i^p),$$

где F_i — множество функций состояния процесса (как активных, так и пассивных); f_i^1 — начальная функция состояния (активная функция), $f_i^1 \in F_i$; $f_{i\,cur}$ — текущая функция состояния, $f_{i\,cur} \in F_i$; T_i^p — текущее время нахождения процесса в текущей функции состояния, или текущее время отсутствия переходов.

Статика процесса определяется элементами F_i и f_i^1, причем для рассмотрения процесса в динамике дополнительно могут использоваться элементы $f_{i\,cur}$ и T_i^p.

Функция-состояние. События и реакция на событие. В свою очередь, j-я функция состояния произвольного i-го процесса описывается парой элементов:

$$f_i^j = \left(X_i^j, Y_i^j\right),$$

где $X_i^j = \left\{x_i^{jl}, ..., x_i^{jL}\right\}$ — множество событий f_i^j; $Y_i^j = \left\{y_i^{jl}, ..., y_i^{jL}\right\}$ — множество реакций f_i^j.

В качестве события функции состояния может рассматриваться произвольная суперпозиция фактов: значение текущей функции состояния некоторого процесса; некоторое определенное значение переменной; некоторое определенное значение текущего времени отсутствия переходов процесса (в виде оператора *timeout*).

Реакция — произвольная суперпозиция действий по изменению значений переменных и текущих функций состояния процессов, определяемая на основе событий текущей функции.

Функции состояния процесса F_i различаются на взаимно непересекающиеся множества активных (F_i^a) и пассивных (F_i^p) функций состояния. Функция состояния процесса пассивна, если его множество реакций пустое

$$f_i^j \in F_i^p : X_i^j = \{\varnothing\}.$$

Среди пассивных функций состояния особо выделены функции нормального останова (f^{NS}) и останова по ошибке (f^{ES}), одинаковые для всех процессов.

Математическая модель событийного автомата. Совокупность совместно функционирующих процессов образует событийный (процессный) автомат — надстройку над процессами. Модель управляющего алгоритма в виде событийного автомата терминологически ориентирована на программную реализацию и позволяет отразить свойства алгоритмов управления: открытость; событийность; цикличность; синхронизм; логический параллелизм.

Событийный автомат представляет собой упорядоченный набор процессов, циклически активизируемых с периодом активизации

$$H = \left(T_H,\, P,\, p_1\right),$$

где T_H — период активизации; P — множество процессов, $P = \{p_1, \ldots, p_M\}$, M — число процессов; p_1 — начальный процесс, $p_1 \in P$.

По запуску автомата текущая функция состояния выделенного процесса p_1 — начальное состояние, для всех остальных процессов — состояние нормального останова

$$f_{1\,cur} = f_1^1,\ \forall_l \neq 1 \Rightarrow f_{i\,cur} = f^{NS}.$$

Модельная сцена. Кроме логического автомата, входными данными для разработки тренажера также является трехмерная модельная сцена. Она должна содержать иерархию структурных объектов и временную дорожку анимации подвижных элементов сцены.

На первом этапе подготовки модели в сцене определяются события, команды и инструменты, воздействующие на объекты модели.

Инструмент — логический элемент, характеризующий режим функционирования модели и обозначаемый уникальным названием и видом иконки.

Событие — логическая единица, определяющаяся выбранным инструментом, названием и объектом трехмерной сцены. При взаимодействии пользователя с трехмерной моделью с помощью требуемого инструмента инициируется данное событие. События в текущей реализации подразделяются на интерактивные и информационные.

Интерактивные события идентифицируются базовым набором атрибутов и используются для конструирования нелинейного сценария взаимодействия пользователя и модели. Информационные собы-

тия расширяют базовый набор атрибутов ссылкой на иллюстративные справочные материалы. При инициировании таких событий в интерфейсе пользователя отображаются описания данных об особенностях поведения объектов.

Команда — логическая единица, имеющая название. Команды, как и события, бывают анимационными и информационными.

Анимационные команды расширяют базовый набор атрибутов именем объекта трехмерной сцены и временным интервалом запускаемой анимации для данного объекта. Информационные команды расширяют базовый набор атрибутов ссылкой на некоторые справочные материалы. При инициировании этих команд пользователю выводится информация в соответствии с текущим состоянием модели.

Второй этап подготовки тренажеров — создание сценариев взаимодействия с пользователем (для игр сценарии являются подобием уровней, для обучающих программ-тренажеров — упражнениями).

Сценарий взаимодействия с моделью удобно выполнить в виде конечного автомата. Это имеет свои достоинства и ограничения. Концепция и архитектура модели предполагает возможность замены механизма сценария взаимодействия для дальнейшего развития функциональности.

Конечный автомат конструируется в виде ориентированного графа состояний, имеет одно начальное и несколько конечных состояний. При инициализации модели сцена приводится в необходимое состояние набором интерактивных команд.

Состояние характеризуется названием, набором отслеживаемых событий и переходами по этим событиям. При инициировании события, указанного в списке отслеживаемых в конкретном событии, осуществляется переход из данного состояния в требуемое. Переход — набор интерактивных и информационных методов. При переходе методы инициируются — запускается анимация и отображается справочная информация в соответствии с текущим состоянием.

Структура программного решения. Предлагаемая концепция реализации предполагает создание двух функциональных блоков: блока взаимодействия с пользователем и блока конструирования интерактивной модели (рис. 1).

Программная структура блока взаимодействия с пользователем выполнена по трехзвенной архитектуре (рис. 2), содержащей:

— информационную подсистему;

— программный модуль модели;

— мультимедийный модуль отображения.

Для создания интерактивной модели первоначально необходимы 3D-модели и анимация некоторых составных объектов, описание методов запуска анимации и событий, возбуждаемых при осуществлении пользователем воздействий на области органов управления модели. Такой формат описания необходим для последующего определения поведения сцены при конструировании сценария упражнения с возможностью вариации (например, переход в «сломанное» состояние).

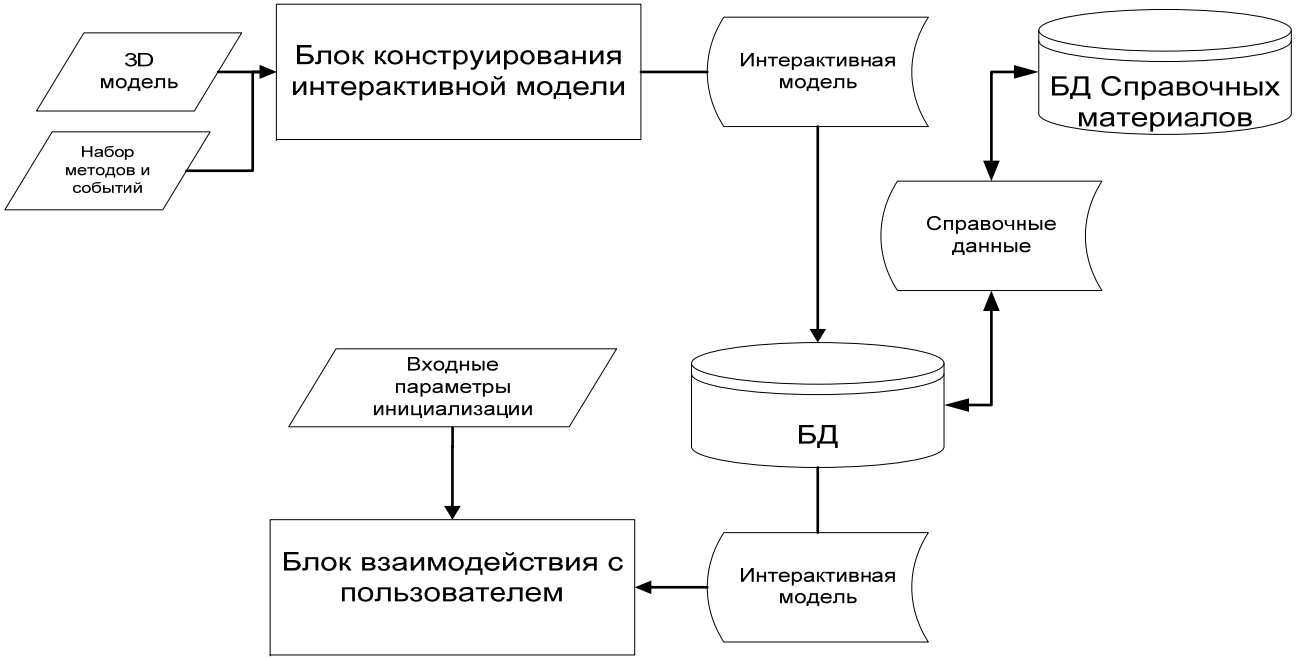

Рис. 1. Схема потока данных (БД — база данных)

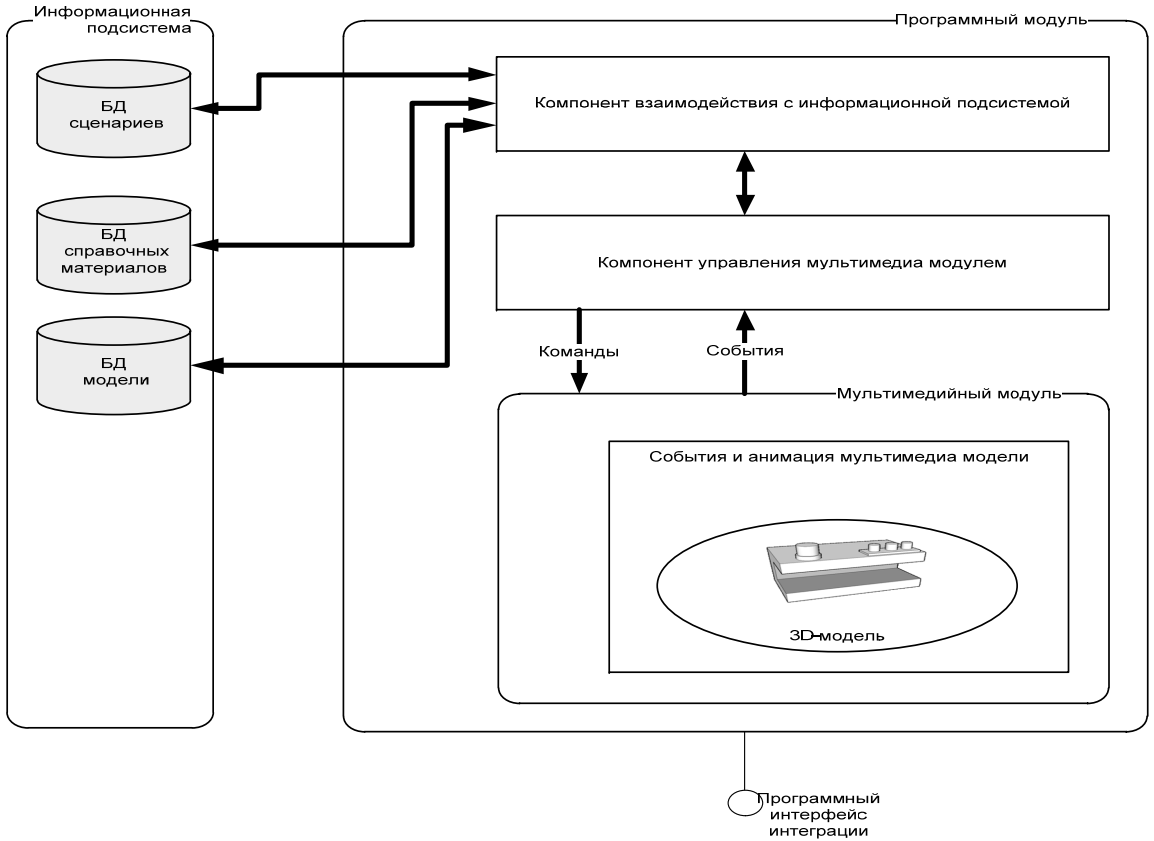

Рис. 2. Программная структура блока взаимодействия с пользователем

Модель и анимация могут создаваться в различных системах разработки 3D-моделей, например 3dsMax или Blender. Для реализации интерактивности использована программная технология Microsoft XNA ver. 3.1 на базе платформы Microsoft .NET Framework ver. 3.5 [4].

Сценарии взаимодействия выполняются в виде конечного автомата после загрузки и обработки трехмерной 3D-модели, а также набора методов и событий.

Переход между состояниями осуществляются по событиям из списка событий. При переходе из состояния в состояние вызываются определенные методы анимации. Сценарий описывается на языке XAML и реализуется на основе технологии Windows Workflow Foundation (WWF) [1]. Пример создания сценария приведен на рис. 3.

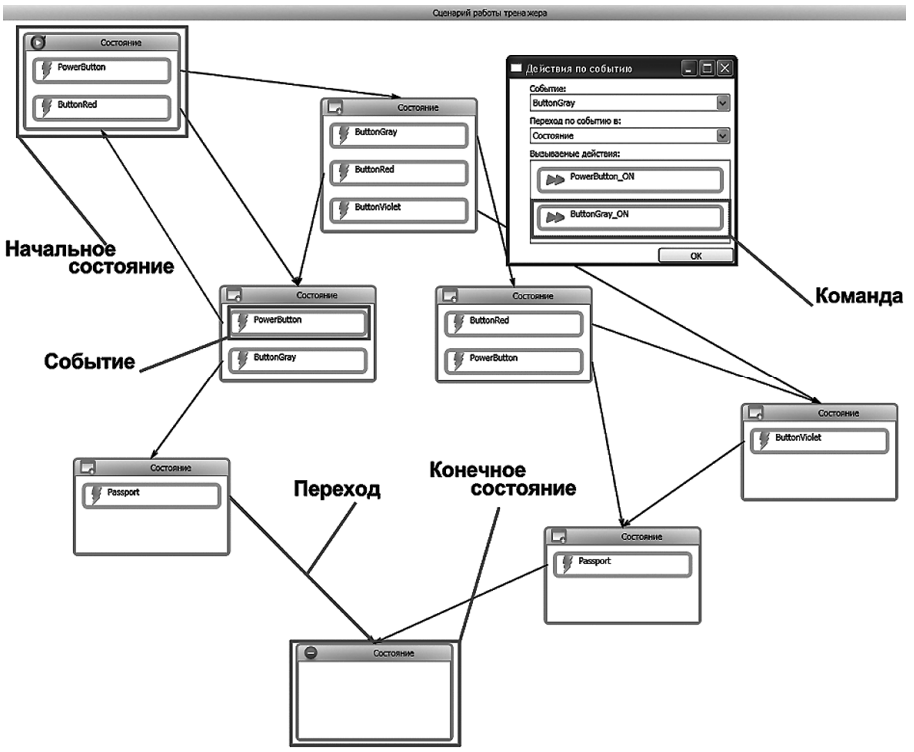

Рис. 3. Пример создания сценария

После обработки 3D-модели и создания сценариев интерактивная модель сохраняется в базе данных упражнений, а сопроводительные справочные материалы — в базе данных справочной информации.

Заключение. Предложенный подход позволяет реализовать законченный инструментарий создания обучающих трехмерных тренажеров. Для создания новых вариантов тренинга на базе модели пользователю не требуется знать языки программирования, достаточно уметь обращаться с пользовательской средой создания трехмерных моделей и анимации. Описание команд и событий, а также

конструирование сценариев являются контекстно понятными в рамках модели действиями пользователя. Для создания сценариев возможен переход от формата конечных автоматов к другим форматам описания поведения системы.

СПИСОК ЛИТЕРАТУРЫ

1. Trowitzsch J., Zimmermann A. Using UML State Machines and Petri Nets for the Quantitative Investigation of ETCS. Электронный ресурс (http://www.zemris.fer.hr/predmeti/fpors/pred/Notes/Petri.pdf).
2. Черненький В.М. Псевдоязык описания сцепленных процессов. Электронное учебное издание. http://db.inforeg.ru/deposit/Catalog/mat.asp?id=285730 (Опубликовано http://iu5.bmstu.ru/nir.php).
3. Bukovics B. Pro WF: Windows Workflow in .NET 3.5 (Expert's Voice in .NET). — NY: Apress, 2008. — 856 p.
4. Reed A. Learning XNA 3.0. USA: O'Reilly Media, Sebastopol, CA, 2008. — 528 p.

Статья поступила в редакцию 6.07.2012